Abstract

This dissertation studies and explores the potential of using Genetic Algorithms to find the shortest path in Vehicle Routing Problem. Routes for vehicles with known location from source to destination are designed where the total distance or time traveled is minimized. Although genetic algorithms have been used to solve the Vehicle Routing Problem this research aims to create different a way of chromosome representation and introduction of overlapping rate to determine the number of chromosome sent for cross-over and mutation. Fixed length chromosomes and their genes have been used for encoding this problem. The proposed method uses proportional selection with two fixed point crossover and random mutation. and discards the infeasible chromosomes by giving high penalty values to the fitness function. The chosen method for crossover and mutation together is shown to improve the rate of convergence as compared to the Djikstra's algorithm. Experimental results show that Genetic Algorithm is able to find a near optimal solution for the sample data in our study.

Keywords : Vehicle Routing Problem, Shortest Path and Genetic Algorithm

Acknowledgement

I would like to thank Assoc. Prof. Raja Noor Ainon, my supervisor for her continuous encouragement, valuable ideas and lengthy discussions. It goes without saying that this thesis would not have reached this level if it were not for the immerse efforts and time spent throughout the course of this research.

Extreme gratitude are due to Mr. Raja Theva Krishnan, for his valuable time and assistance in providing data and information necessary for this research's case study and appreciation to the staffs of Computer Science, Ms. Pappu Sapani, Ms. Emily and Ms. Wan Saripah for their management assistance and advise.

I would like to take this opportunity to thank few of my friends who has contributed towards my success; they are Yeek, Murugesh Rao, Kesavan Krishnasamy, Kannan Ramayah, Sivaneswary Veeramale, Shobana, Marina Tahar and other close friends who have been supportive throughout the course of this dissertation. I would also take this opportunity to thank my bosses Ahmad Albakree and Steven Leong who have been supportive towards my studies.

My greatest thanks go to my beloved family members for encouraging and supporting me in all means possible throughout the course of my master's completion.

Most importantly, my gratefulness to the great Almighty, for bestowing me with all necessities to complete this dissertation successfully.

UNIVERSITI MALAYA
PERAKUAN KEASLIAN PENULISAN

Nama : Shamini Nagaratnam
No.KP:
No. Pendaftaran/Matrik: WGC020061
Nama Ijazah : Sarjana Kejuruteraan Perisian
Tajuk Kertas Projek/Laporan Penyelidikan/Disertasi/Tesis("Hasil Kerja ini"):
"Vehicle Routing Problem Using Genetic Algorithm"

Bidang Penyelidikan: "Artificial Intelligence"

Saya dengan sesungguhya dan sebenarnya mengaku bahawa:

1. Saya adalah satu-satunya pengarang/penulis Hasil Kerja ini;
2. Hasil Kerja ini adalah asli;
3. Apa-apa penggunaan mana Hasil Kerja yang mengandungi hakcipta telah dilakukan secara urusan yang wajar bagi maksud yang dibenarkan dan apa-apa petikan, ekstrak, rujukan atau pengeluaran semula daripada atau kepada mana-mana hasil kerja yang mengandungi hakcipta telah dinyatakan dengan sejelasnya dan secukupnya dan satu pengiktirafan tajuk Hasil Kerja tersebut dan pengarang/penulisnya telah dilakukan di dalam Hasil Kerja ini;
4. Saya tidak menpunyai apa-apa pengetahuan sebenar atau patut semunasabahnya tahu bahawa penghasilan Hasil Kerja ini melanggar suatu hakcipta Hasil Kerja lain;
5. Saya dengan ini menyerahkan kesemua dan tiap-tiap hak yang terkandung di dalam hakcipta Hasil Kerja ini kepada Universiti Malays ('UM') yang seterusnya mula dari sekarang adalah tuan punya kepada hakcipta di dalam Hasil Kerja ini dan apa-apa pengeluaran semula atau penggunaan dalam apa jua bentuk atau dengan apa juga cara sekalipun adalah dilarang tanpa terlebih dahulu mendapat kebenaran bertulis daripada UM;
6. Saya sedar sepenuhnya sekiranya dalam masa penghasilan Hasil Kerja ini saya telah melanggar suatu hakcipta Hasil Kerja yang lain sama ada dengan niat atau sebaliknya, saya boleh dikenakan tindakan undang-undang atau apa-apa tindakan lain sebagaimana yang diputuskan oleh UM.

Tandatangan Calon: Tarikh :

Diperbuat dan sesungguhnya diakui di hadapan,

Tandatangan Saksi: Tarikh :

UNIVERSITY OF MALAYA
ORIGINAL LITERARY WORK DECLARATION

Name of Candidate : Shamini Nagaratnam
IC No : A 35853636 (Old) 770419-14-5980 (New)
Metric No : WGC020061
Name of Degree : Masters in Software Engineering
Title of Project Paper/Research Report/Dissertation/Thesis ('this Work') :
Vehicle Routing Problem using Genetic Algorithm

Field of Study: Artificial Intelligence

Do solemnly and sincerely declare that:

1. I am the sole author/writer of this Work;
2. This Work is original;
3. Any use of any work in which copyright exists was done by way of fair dealing and for permitted purposes and any except or extract from, or reference to or reproduction of any copyright work has been disclosed expressly and sufficiently and the title of the work and its authorship have been acknowledged in this Work;
4. I do not have any actual knowledge nor do I ought reasonably to know that the making of this work constitutes an infringement of any copyright work;
5. I hereby assign all and every rights in the copyright to the Work to the University of Malaya ("UM"), who henceforth shall be owner of the copyright to this Work and that any reproduction or use in any form or by any means whatsoever in prohibited without the written consent of UM having been first had and obtained;
6. I am fully aware that is in the course of making this Work, I have infringed any copyright whether intentionally or otherwise, I may be subject to legal action or any other action as may be determined by UM.

Candidate's Signature : Date :

Subscribed and solemnly declared before

Witness's Signature : Date :

Name :
Designation :

Table of Contents

Table of Figures

List of Tables

List of Abbreviation

AI	Artificial Intelligence
CO	Classical Optimization
EA	Evolutionary Algorithm
EC	Evolutionary Computing
EP	Evolutionary Programming
GP	Genetic Programming
NP	Nonlinear Programming

**Nonlinear Programming (NP) means that it cannot be represented with mathematical model (set of mathematical equation) which we can solve/get a result.

Plateau	Areas where the fitness has small variability
PRCW	Pair Road Connection Weight
RBML	Rule Based Machine Learning
RNCW	Road Not Connected Weight
SGA	Simple Genetic Algorithm
SDLC	System Development Life Cycle
SHAVRS	Sham Vehicle Routing Simulator
SP	Shortest Path
TS	Tabu Search
TSP	Traveling Salesman Problem
VRP	Vehicle Routing Problem

Chapter 1

Introduction

1.1 Background to the Study

Transportation is a major concern in areas which involves logistics. Logistics is an area that incurs high cost, therefore, raises a need for more efficient logistical planning for most manufacturing and distribution industry. Defining the optimal routes for a vehicle in the context of distance traveled and time taken is crucial.

Vehicle Routing Problem (VRP) is a collection of routes that are connected together. Routing decision is based on the shortest path from source to designated destination and returns the origin (source).

(Gendreau, 1998) mentioned that interest in the Vehicle Routing Problem grew rapidly after World War Two, with the increase in postal traffic, catalog ordering of goods from a remote retail, fleet management, facility location, traffic assignment, air traffic control etc.

Genetic Algorithm is known to be able to solve complex optimization problems and it is a field that has attracted many researchers since it is applicable in many areas. In the context of the general optimization problem for routing, a method is needed to compare routes based on their weights to select the shortest path. Genetic methods are well suited to address the route discovery and selection problem. . The potential of GA for solving the shortest path problem has been studied and the results are very encouraging. (Gen,

1997) and (Mohamed, 2000) mentions that GA can find the known optimum very rapidly with very high probability.

1.2 Statement of the Problem

There are many Vehicle Routing systems being used in the fleet management industry. However in many times, road condition that would determine the speed of the vehicle traversing on the road is not taken into consideration. In this dissertation we consider the road condition that would effect the time traveled on the road besides the distance traveled. Road condition can be traffic congestion, road construction or road state (pot holes, uneven, slopes) that would give effect on the speed traversed on the road. Plus, introduce a different approach of GA representation to solve the shortest path determination. The purpose of this dissertation is to develop a VRP simulation using GA and prove in testing that GA does provide an optimum solution for the shortest path.

1.3 Objectives of the Study

1.3.1 General Objective

The first objective of this dissertation is to complete a Vehicle Routing Simulator with good user interface properties that could be used to demonstrate the potential of using genetic algorithm to solve the Vehicle Routing Problem in providing the shortest path. The second objective of this dissertation is to propose a heuristic search method based on genetic algorithm to the Vehicle Routing Problem.

1.3.2 Specific Objectives

What the author hopes the research will find/show:

- To critically review literature related to Genetic Algorithm and how it is used and represented in VRP.
- To develop a simulation program for Vehicle Routing.
- To verify that it is possible to use Genetic Algorithm for solving Shortest Path Problem thus promisingly can be applied for Vehicle Routing Problem.
- To also implement another algorithm (Djikstra) that can be used to prove the correctness of the GA result.
- To develop implement algorithm (Full Search) to display the strength of GA based on computational time.
- To identify ways of selection mechanism and reproduction operators can be manipulated in the process of finding the best solution using Genetic Algorithm.

1.4 Scope of Dissertation

The dissertation is limited to finding the shortest path from source to destination. A stand alone simulation is proposed. Besides the Genetic Algorithm; Djikstra Algorithm and Full Search Algorithm has been implemented into the simulation. Djikstra is used to prove the correctness of the Genetic Algorithm results while Full Search is developed to do a comparison on computation time against genetic algorithm. The SHAVRS has the properties to set the speed for roads on the map and also produce a Fitness Graph for Genetic Algorithm.

1.5 Research Methodology

As for the *literature review*, the Internet has been a valuable resource for obtaining information about this topic especially the IEEE and ACM websites.

As for the *Requirement identification,* some of the information has been collected from relevant journals and books from library. Those references are cited at the reference list of this dissertation.

As for the *Implementation*, the prototype needs to comply with the features based on the requirements that had been identified. A suitable architecture is proposed to support the SHAVRS refer chapter 3:Solution Method. Testing and analysis on the Genetic Algorithm for Vehicle Routing problem was carried out upon completion of the development.

Below is the flow of the research methodology:

1. Literature review on techniques used to solve Vehicle Routing Problem.
    - To acquire understanding and knowledge on other Algorithms used for Vehicle Routing
2. Literature review on how Genetic Algorithm is applied
    - To understand how GA operates.
    - To acquire knowledge on other fields where GA is applicable.
3. Literature review on VRP using Genetic Algorithm.
4. Introduction of Shortest Path in VRP.
    - To understand the use of GA in solving shortest path/route problem
    - To demonstrate the potential of using genetic algorithm to solve the shortest path problem

- To understand how this shortest path in GA can be applied to VRP

5. Propose a GA solution to VRP in finding the shortest path.
6. Develop a GA simulation tool to solve VRP using C#.net
7. Experiments on GA tool developed for VRP
    - Investigate and experiment to demonstrate the potentials of using GA to solve SP for VRP.
    - In order to be able to check the correctness of the GA solution, Djikstra algorithm and Full Search are used as benchmark to verify the GA the results.

1.10 Dissertation organization

This report is organized in the following manner:-

a) Chapter 2 – Literature Review

This chapter provides on overview on the background and theory of the vehicle routing problem and genetic algorithm. It then discusses the related research done using genetic algorithm to solve the vehicle routing problem and it also covers briefly on other existing algorithms used for vehicle routing. Introduces the reader to GA from its history and biological significant, plus, run through the techniques used in GA from chromosome representation, denotation of Fitness value, Selection Mechanism to its Reproduction Operators.

b) **Chapter 3 – Solution Method**

This chapter we propose a new algorithm by extending the conventional GA method which is capable in solving the shortest path for the VRP. Shows how the

chromosome is represented, selection mechanism that was used and the type of operators applied and how fitness value is calculated.

c) **Chapter 4** – **Simulation System Design**

This chapter will give the user a scheme of how the simulation that is developed looks like. It covers the each parameter options on the GUI and the representation of the results.

d) **Chapter 5** – **Results and Analysis**

This chapter presents an experiment to find the optimal parameter setting for the proposed genetic algorithm simulation. This chapter will cover the variables and the constraints in the simulation. It also leverages on the testing details for the genetic algorithm using a simulator, and the results analysis of the routing problem. This chapter also includes discussion on the result and analysis done.

e) **Chapter 6** – **Conclusion**

This chapter concludes the work in exploring the potential use of genetic algorithm to solve the vehicle routing problem with time windows and also covers the future or continuation study or work that can be done to future enhance the research or the simulation that was developed.

1.11 Summary

As a conclusion, this chapter is basically an introduction to the entire dissertation. It covers the problem statement, scope of research, processes and steps taken to complete the system and documentation of the entire dissertation. The next proceeding chapters will take you through the solution method used to develop the SHAVRP, the simulation design and the testing done.

The propose algorithm has good user interface and is developed with properties to control the speed on the road for the VRP.

Chapter 2

Literature Review

2.0 Introduction

In this chapter there are 7 main sections.

Section 2.1 is on introduction to VRP, which covers the importance of VRP and its elements.

Section 2.2 explains the TSP to illustrate the difference between VRP & TSP

Section 2.3 presents the current algorithms that are being used to solve the VRP and briefly takes you through these algorithms to show the difference in approach between other algorithms and GA.

Section 2.4 introduces basic concepts of GA. Also covers its strength and weakness.

Section 2.5 elaborates on shortest path determination and its use in GA.

Section 2.6 discusses the Djikstra algorithm.

2.1 Vehicle Routing Problem

The basic problem class we address in this dissertation is known as the Vehicle Routing Problem (VRP) (Dantzing, 1959) (Toth, 2001), which consists in delivering goods to a set of customers with known demands through minimum-distance and time vehicle routes originating and terminating at the depot, as seen in Figure 2.1 from (Dantzing, 1959).

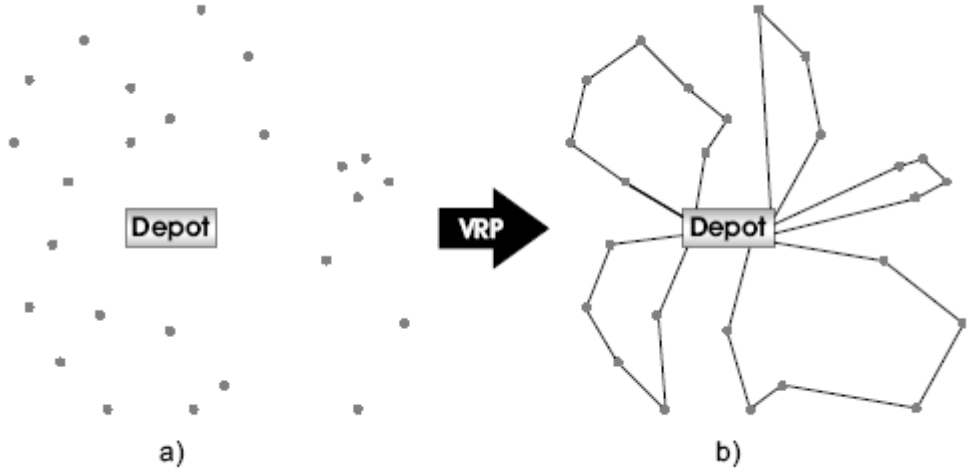

Figure 2.1 : The Vehicle Routing Problem finds the minimum time/distance routes (b) to serve a set of customers (points) geographically distributed, from a depot (a) (Dantzing, 1959)

VRP signifies a fleet of vehicles to supply a set of customers. Each vehicle has a certain capacity and each customer has a certain demand. Further on, there is a depot(s) and a distance (length, cost, time) matrix between the customers. We look for optimal vehicle routes (minimum distance or number of vehicles).

VRP deals with determining least cost routes from a depot to a set of scattered customers. The routes have to satisfy the following criteria:

- Each customer is visited exactly once
- All routes start and end at the depot
- Sum of all demands on a route must not exceed the capacity of a vehicle

The VRP is a heuristic based Nonlinear Programming (NP) problem as mentioned in (Christofides, 1979). Heuristic approach is preferred compared to exact method. (Toth,

2001) has mentioned that no known exact method is capable of consistently solving to optimality instances involving more than 50 customers.

2.1.1 Application of Vehicle Routing Problem

Case 1 : American Red Cross

ARC's blood service division is an exemplary for VRP. ARC's blood collection is carried out at predetermined sites where the collection of the blood has to be done within 6 hours. The blood collected has to be processed to render into blood components such as plasma, platelet, red blood cells and other derivatives. In order to extract platelets, it is important for the whole blood to be processed within 8 hours after donation. The donor's quarantined blood specimen is then tested and distributed to hospitals according to predetermined schedules. A blood division annually spends several million dollars on costs directly related to each operation described above. ARC has to decide from which sites to shuttle blood and how these shuttles should be routed, in order to provide an adequate amount of platelets at minimum costs and within the permitted time.

Case 2 : Delaware Valley Wholesale Florist Inc.

Meeting the delivery of more than 3000 florist in the middle Atlantic States is challenging to Delaware Valley distribution fleet. Since flowers are perishable goods, it has to be delivered fresh. It sustains a constraint in transportation time which begins from the purchasing of these flowers from contracted wholesalers, off shore growers and U.S. based brokers o the customer itself. It is also important that the pick up and delivery schedules would be met in a cost effective manner. Thus, VRP is required here so that the pick up deliveries such as loading, unloading and waiting times from various places are

done timely with cost efficiency. Thus, taking into consideration of traffic congestion, number of sequence stops & available cargo space are taken into consideration so that promised delivery times are met and that flowers are delivered to customers fresh.

2.2 Difference between Traveling Salesman Problem and VRP

Vehicle Routing Problem (VRP) is the *m*-TSP, where a demand is associated with each city or customer and each vehicle has a certain capacity. According to Bullheimer et. al. (Bullnheimer, 1997), as soon as the customers of the VRP are assigned to vehicles, the problem is reduced to several TSPs.

In **Traveling Salesman Problem** (TSP) a number of cities have to be visited by a salesman who must return to the same city where he started. In solving the problem one tries to construct the route so that the total distance traveled is minimized. In the *m*-TSP problem, the *m*-salesman has to cover the given cities and each city must be visited by exactly one salesman. Every salesman starts from the same city, called depot, and must return at the end of his journey to this city again. When the distance is minimized, this would also mean the cheapest way to visit all the nodes and return to the starting point. Cheapest way directly refers to the time taken to travel the distance.

TSP traces its origin to the so-called Icosian Game, invented in the 1880's by the Irish mathematician Sir William Rowan Hamilton, in which a player should find the way to visit all 20 points of a two-dimensional representation of an icosaedar, without visiting any points more than once.

2.3 Types of Algorithm for Vehicle Routing

Based on Fisher (Fisher, 1992), vehicle routing methods fall into three main categories. Simulated Annealing, Tabu Search, Ant System and Genetic Algorithms are algorithms that fall under 'exact optimization' category. The following paragraphs will briefly describes how each exact optimization works. The algorithm used in this dissertation is Genetic Algorithms category. In depth on GA will be discussed in the section 2.4

Simulated Annealing idea is taken from the industrial process called *annealing* (Haupt, 1998, pp.16). This technique was invented by Kirkpatrick 1982. SA only deals with one candidate solution at a time and no information from previous moves are saved to guide the selection of new moves. It is one directional search unlike GA which is multidirectional search.

Glover (Glover, 1986) explains the **Tabu Search (TS)** as a "meta-heuristic superimposed on another heuristic." TS prevents algorithm from going in cycles as such previous solution space will not be checked in the next iteration of solution. Those points are declared 'tabu'. TS first looks for a local minimum, and to avoid repeating the solutions it already examined, it stores them in one or more Tabu lists.

Bullheimer et al. (Bullnheimer, 1997) described the **Ant System** as a distributed metaheuristic for solving hard combinatorial optimization problems. This algorithm was first used to solve the TSP. It was introduced by Colorni, Dorigo and Maniezzo, and is based on observed behavior of real ants' colonies in search for food. Fellow ants can

follow the pheromone trail and while following it they will additionally mark it with new pheromones..In result, paths which quickly lead to rich food sources will be reinforced.

Genetic Algorithm (GA) is an algorithm model based on genetic evolution and is being expressed using genotypes. The original GAs was developed by J. Holland at the University of Michigan in 1975. Since then this conventional GA has been changed, by applying different chromosome representation schemes, selection method, cross-over, mutation and elitism operators. Detail discussion in GA is covered in section 2.4.

2.4 Genetic Algorithm

Genetic Algorithms (GAs) uses biology evolution based on Darwin's theory such as inheritance, mutation, natural selection and recombination (or crossover). In GA population is a set of solution or individual instead of chromosomes, crossover operator plays a role of reproduction from parent solution to provide offsprings. Mutation is assigned to make random changes in the solution to produce diversity into the population. Solutions are evaluated based on fitness value. Fitness value is calculated based on the problem of the search.

(Chang, 2002) has presented Uni-casting or one-destination algorithms such as the one proposed in this dissertation. Advantages of using GA have been discussed in section 2.4.7. The potential of GA for solving the shortest path problem has been studied and the results are very encouraging. (Gen, 1997) provides idea of constructing effective solution for shortest-path based problems. The following section 2.4.1 takes us through the

process of applying selection method, fitness function, operators and genetic representation into GA.

2.4.1 Working Mechanism of GAs

GA starts with a set of chromosome referred to as ***initial population*** of individuals which are generated randomly or heuristically. If an initial population is randomly generated, it would take longer time for the algorithm to converge to the solution whereas if it is generated through some heuristics it makes the chromosomes in the population closer to the solution; hence taking less time to converge. GA works on the coding of the solution instead of the solution themselves, which is why an efficient representation of chromosome is needed.

A ***selection method*** will be used to select prospective parents from the population for recombination and survival. Proportional, ranking and tournament are few examples of selection methods. It is known that Selective pressure has an influence on convergence, (refer to section 2.4.6).

Quality of a solution is measured through ***fitness value*** which enables them to be compared so that the highest fitness solutions are taken into next generation. Fitness function is another character of GA that has to be taken care when defining the calculation for fitness because it determines the quality of the next generation.

Creation of new members is done by ***crossover and mutation operations*** so that new population will be better than the previous one. Discussion on this genetic operator is

covered in section 2.4.2. This operations is repeated until some conditions are satisfied like predefined number of generation is produced or when there is no more improvement in the population. Refer section 2.4.2 for terminating conditions.

Genetic algorithms typically have the following structure, figure 2.2

```
initialize timer
generate a random population
evaluate fitness
        while not terminated do
        {
                increment timer
                select the fittest parents
                recombine genes of selected parents
                introduce mutations
                evaluate fitness
                select survivors that will become the next generation
        }
```

Figure 2.2 : Basic Pseudocode for Conventional GA

During the first three decades, researchers assumed that GA cannot be used to solve problems but (Fogel, 2000) says that it is not due to GA but due to limited computing power during that time.

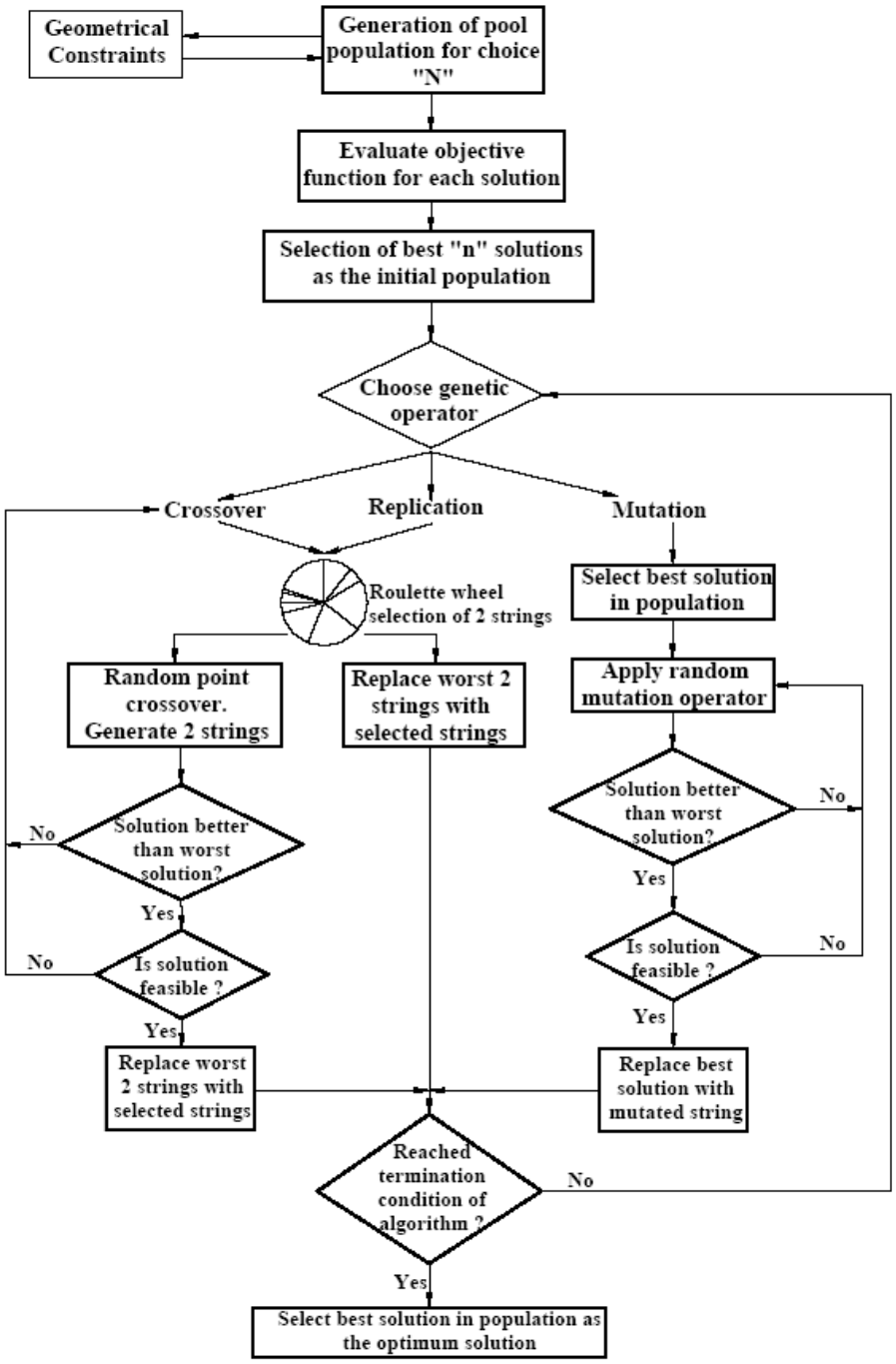

Figure 2.3 : Basic Genetic Algorithm Flowchart
(Al-Tabtabi, 1998)

2.4.2 Genetic Operators of GA

a) Replication

Two random solutions are chosen to replace the worst two solutions in the population. There is no need to check for the feasibility of the solutions, as no modifications were performed (crossover, mutation).

b) Crossover

The same selection procedure is applied to select the parents that will be crossed. There are many different crossovers like simple, random, 2 point or multi point. Simple single-point crossover was used so as to minimize the disruption of the schemata.

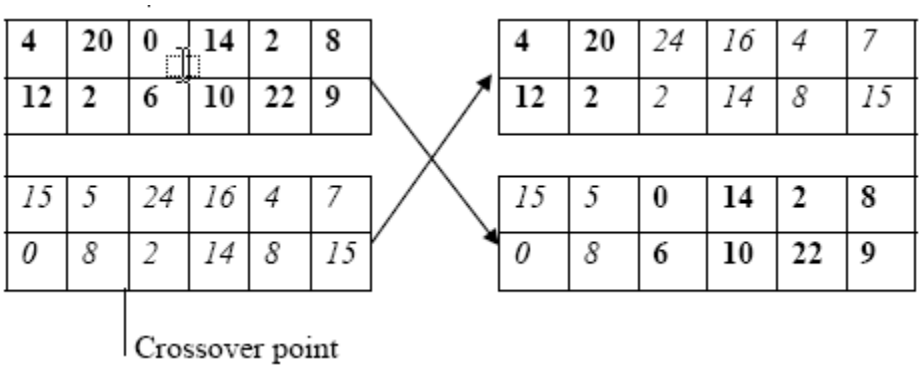

Figure 2.4 : Single Point Crossover

After the crossing is done, two checks are performed on the offspring which are Constraint satisfaction to check the feasibility of the offspring and Objective function improvement is to verify that the new solution is not worst than those being replaced.

c) Mutation

Mutation is done after crossover. It is a swap of gene sequence in the chromosome to break current stagnation thus introducing new genetic information into the population. It

was noticed during the testing of the system that performing the GA without mutation led to solutions that required slight refinements to reach more optimum solutions. These refinements usually involved very small movements of one or more facilities in a specific direction.

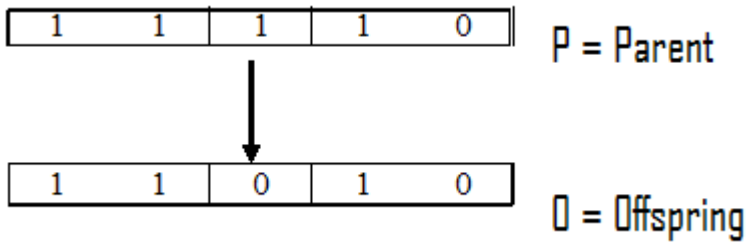

Figure 2.5 : Illustration of simple mutation

The mutation operator randomly chooses the following: Which gene to be moved, whether the movement should be in X or Y direction and if it should be a +ve or –ve direction. It then applies a movement of 1 unit to the facility chosen and in the direction chosen. New solution is checked for its fitness value for improvement. If not met, the mutation procedure is repeated. A simple version of mutation is showed in figure 2.5 that bit number 3 was chosen randomly to mutate thus the offspring gene has changed from 1 to 0.

d) Terminating Conditions

Until a termination condition is reached, the generation process is repeated. Common ***terminating conditions*** are when fixed number of generation is reached, when allocated budget for computation time or money is used up, when an individual that satisfies the minimum criteria is found, when highest ranking individual is reached or when better results are not produced by successive iteration or any of the combination above.

2.4.3 Convergence Condition

Convergence is the movement towards consistency. (DeJong, 1975), quoted by (Robert, 1997), was that a gene is said to be converged when 95% of the population share the same value. However, GAs are stochastic iterative process therefore not guaranteed to converge; hence, the ***termination condition*** (refer section 2.4.2) may be specified.

In section 2.4.1 we discussed the selection methods for GA. Selective pressure does influence convergence of the algorithm. Too fast improvement mean weaker individual are dropped from population too soon so their characteristics are not passed on. Weak selective pressure makes search ineffective by choosing weak individuals so it is critical to balance off this selective pressure to get good solution.

Figure 2.6 shows how fitness varies in a typical GA. As the population converges, the average fitness will approach that of the best individual.

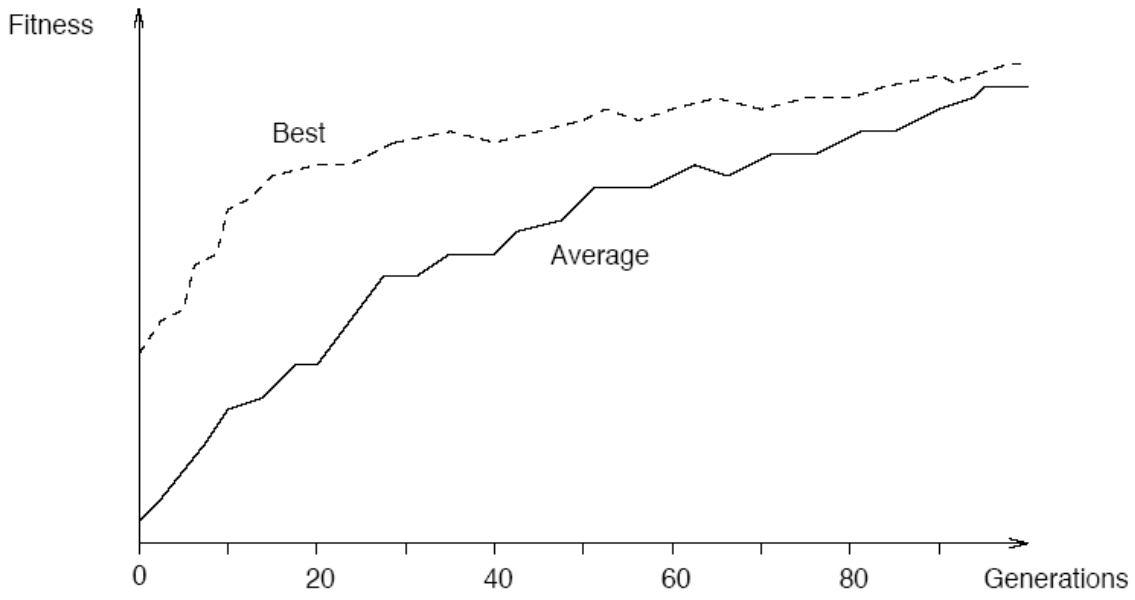

Figure 2.6 : A typical GA Run
(Mitchell, 1996)

2.4.4 Conditions when GA Algorithm is Chosen

GAs are good at finding "acceptably good' solutions "acceptably quickly". If the parameters discussed in section above are not defined well then it is not guaranteed that GA will find global optimum solution to a problem. (Garey, 1979) suggests that GA should be when there is no other better problem solving strategy and the problem domain is NP-Complete.

The usage of GA as suggested by (Al-Tabtabi, 1998) in the following situations:

1- Conventional statistical & mathematical methods are inadequate.

2- The problem is very complex, because the possible solution space is too large to analyze in finite time.

3- Conventional methods are not practical as the additional information given to guide the search is not available or not enough.

4- The solution can be obtained by using form of strings and characters to encode the problem.

5- The problem is huge and not readily understood.

6- Near optimal solutions are needed to be used urgently as starting points.

2.4.5 Strength of GA

Multiple Direction

Most other algorithms are serial. They search for solution to a problem space in one direction at a time. If the solution they found turns out to be not the best, they would abandon all work previously taken and start over. Since GAs have multiple offspring, it can explore the solution space in multiple directions at once. If one path turns out to be a

dead end, they can easily eliminate it and continue work on more promising path. This gives GA a greater chance of finding optimal solution in each run.

No foresight about the problems they are set to solve

GA makes *random* changes to their candidate solutions and then uses the fitness function to determine whether those changes produce an improvement. GA does not use previously known information and do changes to it for improvement, Decisions on solution are based on randomness, therefore all possible search pathways are theoretically open to a GA.(Goldberg, 1989 p.23] says that GAs do not have problem of misinterpretation based on established beliefs of "how things should be done" or what "couldn't possibly work".

Exploitation and Exploration

Optimization algorithms must be able to balance the difference between exploration and exploitation. A classical algorithm focuses on exploration of the search space but neglects exploitation, whereas a random search algorithm concentrates on exploitation but neglects exploration. GA has proven to explore solution space for new solution while exploiting already visited solutions.

2.4.6 Limitation of GA

Fitness function

To get better solution for a given problem, fitness function must be carefully determined to attain higher fitness. If the fitness function is chosen poorly or defined inaccurately, GA either will not find a solution to the problem, or may end up solving the wrong problem. Although in reality, the GA is doing what it was told to do in the algorithm but

it is not what its creator had in mind for it to solve. An example of this can be found in (Graham, 2002).

Parameters of a GA

Besides making a good choice of fitness function; GA parameters such as type of selection process, population size, rate for mutation and crossover has to be decided with attention. If the population size is too small, GA will not have enough solution space to explore to find good solutions. If the rate of genetic change is too high good chromosome may be disrupted and if the selection scheme is chosen poorly, the population in hand will not be a strong one.

Premature Convergence

This is a well-known problem that can occur with a GA. If a fit individual amongst its competitors emerges early on in the course of the GA run, this would bring down the population's diversity too soon, leading the algorithm to converge on the local optimum that this individual represents rather than searching the fitness landscape thoroughly enough to find the global optimum. (Forrest, 1993 pp. 876) mentions that premature convergence happens commonly to small populations.

2.5 Shortest Path

The shortest path problem is concerned with finding the shortest path from a specified starting node (source/origin) to a specified ending node (destination) in a given map while minimizing the total cost associated with the path. The shortest path problem is a classical combinatorial optimization problem having wide spread applications in a variety of settings. The applications of the shortest path problem include vehicle routing in

transportation systems (Bodin, 1983) and traffic routing in communication networks (Topkis, 1991). The shortest path has been investigated extensively in these mentioned researches.

One the most common problems encountered in analysis of vehicle routing is the shortest path problem: finding path between two designated nodes having minimum total length or cost. (Ravindran, 1987) in his study provides an interesting alternative to using genetic algorithms to find out the shortest path. (Ida, 1995) says that shortest path problems form an important base for other relevant problems and is applicable in vehicle routing. Therefore, this study provides base for constructing efficient solution procedures for VRP based on shortest path.

2.6 Djikstra Algorithm

E.W. Djikstra (Djikstra, 1959) initiated and solves the problem of finding a single source shortest path problem from source to a destination by using algorithm called "Djikstra Algorithm". By using this algorithm, we could find the shortest path or single source shortest path to all points in a map at the same time. The graph representing all the routes from one vertex to all the others with a spanning tree concept is related to spanning tree problem. In this situation it had to include all the vertices.

G = (V, E)

Where V = set of vertices & E = set of edges

The algorithm proceeds in stages and can be formally described as follows, first define the variables:

V = set of vertices in the network

s = source vertex

T = set of vertices so far incorporated by the algorithm

w (i, j) = link time from vertex i to vertex j, where

w (i, j) = ∞ if the two vertices are not directly connected

w (i, j) $\geq$ 0 if the two vertices are directly connected

L (v) = cost of the least-cost path from vertex s to vertex v that is currently known to the Algorithm; at the termination, this is the cost if the least-cost path in the graph from s to n

The algorithm has three steps. Step 2 and 3 are repeated until final paths have been assigned to all vertices in the network, where T = V.

Step 1. Initialization

T = {s}

i.e. the set of vertices so far incorporated consists of only the source vertex

L (n) = w (i, j) for n $\neq$ s

i.e. the initial path costs to neighboring vertices are simply the link costs

Step 2. Get next vertex

Find the neighboring vertex no in T that has the least-cost path from vertex s and incorporate that vertex into T. Also, incorporate the edge that is incident on that vertex in T that contributes to the path.

Step 3. Update least-cost path

L (n) = min [L (n), L (x) + w (x, n)]

If the later term is the minimum, the path from s to n is now the path from s to x concatenated with the edge from x to n.

The algorithm terminates when all the vertices have been added to T. Thus, the algorithm requires V iterations. At termination, the value L(x) associated with each vertex x is the cost of the least-cost path from s to x. In addition, T is a spanning tree of the original digraph and defines the least-cost path from s to each other vertex.

2.7 Summary

Djikstra method is a well known algorithm for finding optimum path is shortest path search problems. However, time is required to find optimum solution. Djikstra has poor search efficiency and long search time when the number of nodes increases which makes Djikstra is unsuitable for real time problems.

GA's search time is determined by the number of genes that are produced, therefore is not affected by the size. GA finds solution is much lesser time than Djikstra, although GA might not find the best solution, it can find a near perfect solution. Selection method that was chosen is Proportional selection because the probability of choosing an individual is directly proportional to its fitness value. The larger the fitness value, the individual is more likely to be selected as a parent. For this study the approach chosen is the GA algorithm to find the shortest path for VRP.

Chapter 3

Solution Method

3.1 Chapter Introduction

The prototype developed is named as SHAVRS, an acronym used to denote this simulation. The first 3 alphabets, SHA denotes the authors' name, indicating that this simulation was developed by the author, and VRS stands for Vehicle Routing Simulator.

With the completion of a thorough study on VRP, GA and SP in chapter 2, the method of constructing a vehicle routing solution using genetic algorithm in this simulation is discussed in this chapter. The chapter begins with an overview of the approach taken in this dissertation to create a vehicle routing simulation. Subsequent sections would focus on each major step taken for this approach. This would cover the discussion on the algorithm used for the implementation. Furthermore, pseudo-codes for some GA operators are prepared before actual implementation into the simulator. In addition, some details are explained through sample diagrams and figures, to let reader have a better understanding or visualize the ideas taken.

3.2 Methodology Summary for Genetic Algorithm in Solving Vehicle Routing Problem

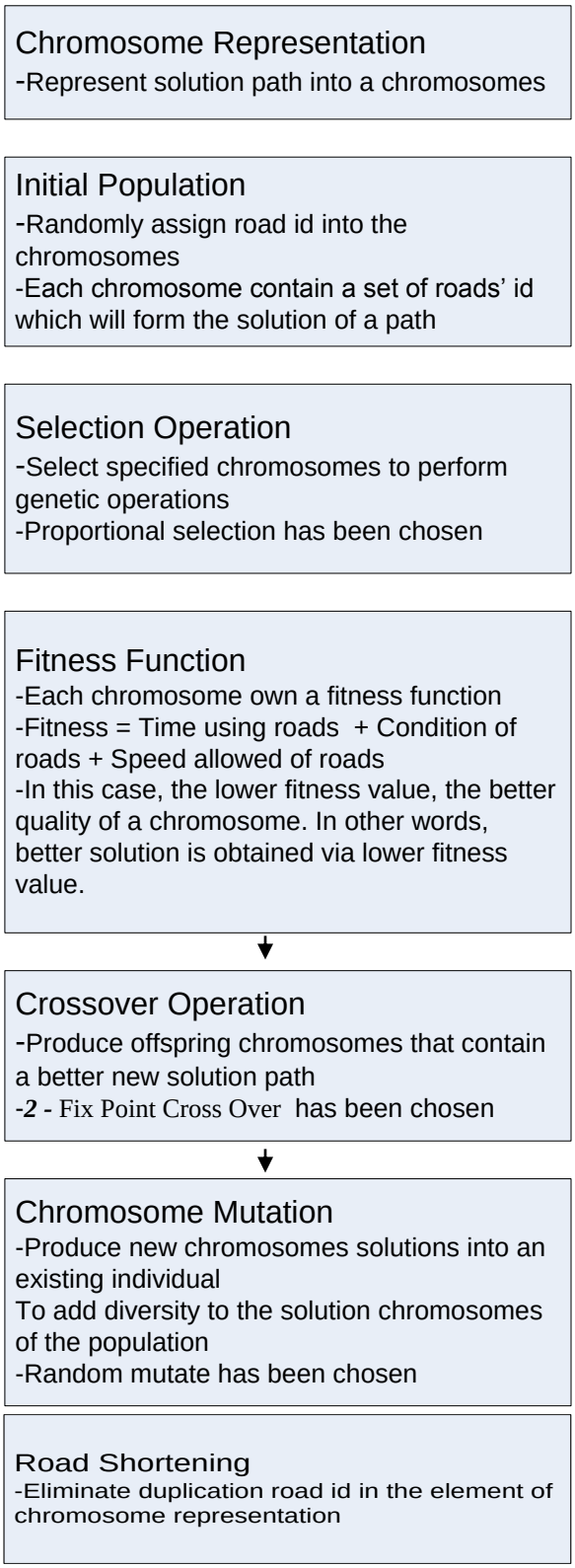

Figure 3.1 : Genetic Algorithm Steps

3.3 Chromosome Representation

r_1	r_2	r_3							r_n

Figure 3.2 : Chromosome Representation

1. There are 15 elements in each array, where n=15.
2. Arrays are of fixed length of 15 elements.
3. Each array represents one chromosome.
4. r_1, r_2, …, r_n is defined as Road ID.
5. Every road is identified by a Road ID.
6. A road chromosome contains a list of elements.
7. Each element contains a Road ID.
8. Elements which has the value is -1 is considered not used, does not have a Road ID.

E.g. r_1, r_2, r_3 only used 3 elements in the array of 1 chromosome; rest of elements in the array is placed with null value, -1.

To get from source r_1 to destination r_5, the route taken passes through 7 road ID which are in r_1 r_7 r_8 r_3 r_9 r_4 and r_5. The road ID are placed in the array such as below figure 3.3 :

r_1	r_7	r_8	-1	-1	-1	r_3	-1	r_9	-1	-1	-1	r_4	-1	r_5

Figure 3.3 : Array to fill Null Value

Since the array consists of 15 elements, and only 7 elements are occupied for the desired source and destination, therefore the balance 8 elements in the array are placed with null value (-1).

9. The element of a chromosome will be filled up with road id that forms a solution path.

10. Element of chromosome will be placed a null value when the road id in this element is found duplicate with other road id contained in other elements. Refer section 3.6.2

11. The cost of using a road is defined as Road Cost, where

    Road Cost = Speed + Distance

    E.g. Array 1, Distance 24, Speed 25

    Array 2, Distance 10, Speed 20

    Road Cost R1 = 24 + 25 = 49

    Road Cost R2 = 10 + 20 = 30

12. The lesser the Road Cost value, the better solution. Cost is the weight given to each road.
13. Therefore, R2 is better than R1.
14. Every road chromosome has its own fitness value, defined as fitness value, where
    (a) Fitnessvalue = The sum of Road Cost for every road in a road chromosome + *Road_Not_Connected_Weight + Pair_Road_Connection_Weight*

(b) *Road_Not_Connected_Weight* (RNCW) is added to the fitness value when the path represented by the specified road chromosome is not connecting the source and destination location.

(c) Constant value for RNCW is 3000.

Example for *Road Not Connected Weight*

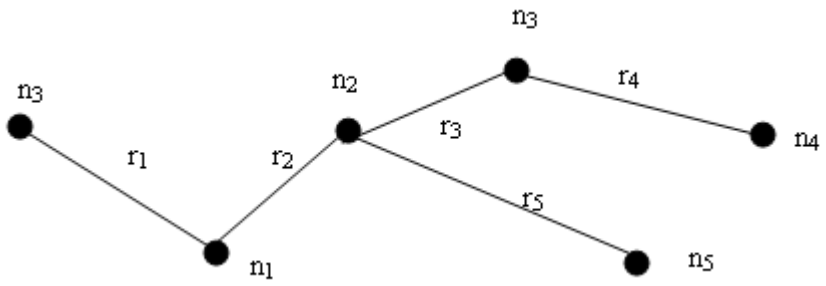

Figure 3.4a : Graphical Representation for Road Not Connected Weight

r_1	r_2	-1	-1	-1	r_5	-1	-1	-1	r_4

Figure 3.4b : Chromosome Representation for RNCW – Not Connected

r_1	r_2	-1	-1	-1	-1	-1	r_3	-1	r_4

Figure 3.4c : Chromosome Representation for RNCW – Connected

Source is N3 and the Destination is N4. Figure 5.4b shows a situation that source and destination is **NOT** connecting while Figure 5.4c shows a condition where source and destination are connected.

(d) *Pair_Road_Connection Weight* (P*R*CW) is a weight that will add to the fitnessvalue when a adjacent road is not connected. Otherwise, a weight will be deducted from the ***road*** fitnessvalue.

(e) Constant value for *PRCW* is 100.

Example for 2 **NOT** adjacent roads:

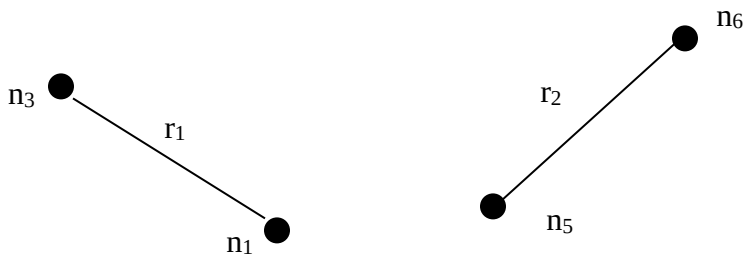

Figure 3.5a : Graphical Representation of NOT adjacent roads

r_1	r_2	r_3	r_4						r_n

Figure 3.5b : Chromosome Representation of NOT adjacent road

Road r_1 and r_2 are **NOT** adjacent to each other; therefore P*R*CW will be added to ***road*** fitness value.

Example for 2 **adjacent** roads:

r_1	r_5	r_3	r_4						r_n

Figure 3.6a : Chromosome Representation of 2 Adjacent Road

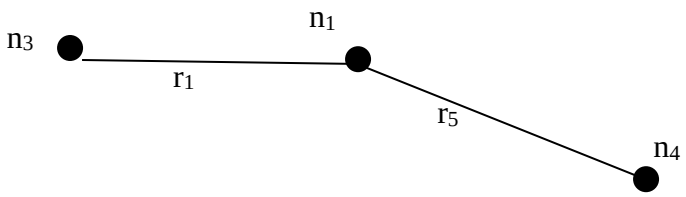

Figure 3.6b : Graphical Representation of 2 Adjacent Road

Road r_1 and r_5 are **adjacent** to each other, therefore *PRCW* will be **deducted** from ***road*** fitness value.

Note that these 2 adjacent roads will share a same location, which is $\mathbf{n_1}$.

(e) The smaller the 'fitnessvalue' value, the stronger road chromosome is obtained.

(f) Every chromosome is initialized as the roads which contain the source location and the destination location are fixed at the start and end of the array respectively.

15. Each chromosome is a solution *path*.

16. Distance or Length of road is calculated based on the formula :

$$D = \sqrt{(x_2 - x_1)^2 + (y_2 - y_1)^2}$$

3.4 Simulation Representation

1. Nodes = Location

2. Edge = No routes

3. Population: Number of Chromosomes

E.g 500 population = 500 Chromosomes. It will generate 500 possible solution paths**.**

4. Crossover Rate: How often for the chromosome to perform cross over operations.
5. Mutation Rate: How frequent changes of road id in one element for a specified chromosome.
6. Overlapping Rate: How many are sent for crossover and mutation from the entire population. (*Refer section 3.6.3, Figure 3.7***)**
7. Number of generation: Number of iteration, how many runs to get final result.
8. Best solution would have minimum path value or minimum distance therefore shorter time. Also, fitness chromosome value.
9. Fitness Value = Best Solution (as explained in point 8)
10. Speed = Length / Time

    Where Length = km=kilometer

    Time = s = seconds

    Therefore, Speed = kms^{-1} = km / s
11. Heuristic is used in the Initialization phase. In order to get a faster convergence (solution path), the potential road that would connect the Source and Destination is filled in the chromosome. Roads IDs that would not contribute in the link of Source and Destination would not be used to occupy the chromosome.
12. Cost Value of a road is determined by Speed and the Distance.

3.5 Simulation Assumption

1. Road on map does not consist on any crossover roads,
2. Array size is 15 elements, to fix the chromosome length.
3. Road id which contains the Source and Destination location in the array element

remains in array without any GA operators done on it.

****** As such the above assumption/constraint builds an offspring by preserving edges from both parents. Edges in this case would be the source and destination road id.

4. No direction. Not a directional map. E.g. N12 – N7 is the same as N7 – N12.
5. Locations or Road id position on the map are randomly generated. It is on real time and is stored in data structure.

3.6 Genetic Algorithm Used

3.6.1 Pseudo Code

1. Initialize populations

2. Set g = 0 where g = number of population.
    a. While (g still less than proposed generation)
        i. Select c1 chromosome using Proportional Selection Algorithm
        ii. Select c2 chromosome using Proportional Selection Algorithm
        iii. Perform Two-Point Cross-Over Algorithm to the selected chromosomes c1 and c2
            1. Let assume *I* is the size of the chromosome.
            2. Compute two random variable e1, e2 where 1<=e1, e2<=*I*
            3. $m_i = 0$ for all i=1, …, *I*
            4. For each i=e1, … , e2, let $m_i = 1$

5. Refer to mask vector m, swap chromosome value at index i where $m_i = 1$.

iv. Sort the population chromosome based on fitness value before Mutation

v. For population index which more than *(*Overlapping Value*/*100)*Population, perform Random Mutate operation.

1. For each i=1, …, *I*

a. Compute a random value, e, where 0<e<1

b. If e <= mutation rate then replace the old element value at index i with a new generate value.

vi. Increase generation counter, g.

vii. Sort the chromosome set based on their *fitnessvalue*.

viii. Repeat step (a).

3.6.2 Explanation for the pseudo code

1. Let assume *I* is the size of the chromosome.
2. Compute two random variable e1, e2 where 1<=e1, e2<=*I*

I is the length of the chromosome, in our simulation, it is set to 15. Therefore I = 15. e1 is the 1st element and e2 is the 2nd element chosen from the chromosome. e1 should be more or equal to 1 where else e2 has to be lesser or more than 15.

3. $m_i = 0$ for all i=1, …, *I*
4. For each i=e1, … , e2, let $m_i = 1$

Refer to mask vector m, swap chromosome value at index i where $m_i = 1$.

m is the mask vector.

For example:
Point 5 and 10 is chosen from the chromosome is taken for crossover. The mask would set as such that m_1 to m_4 is 1 and m_5 to m_{10} is set to 1 while m_{11} to m_{15} is set to 0.
The program would search for m_i with value 1 to do a crossover, and continue in loop until m_i is no longer of value 1.

After the first crossover, the population is sorted, to push the good ones above and the bad ones below so that the mutation is done to the bad chromosomes to see if it can made into better solution path.

3.6.3 Road shortening

1. Check array for Duplicate road, replace the second duplicate road with null value (-1).
2. Chromosome = array
3. Inside each array is stored with road id.
4. *Refer Section 3.6.3*, *Figure 3.8*
5. This would assemble a chromosome that does not have duplicate road id. Consequently, gives a solution path that does not contain a duplicate road id.

3.6.3 Graphical Representation

1. Population Size : 100 (100 chromosome/array).
2. Initial population is not sorted.

N_a								N_b

Figure 3.7 : Chromosome Sample 2

3. N_a is the Road Id which contains the source location while N_b is the Road id that contains the destination location which is chosen from the Vehicle Routing Properties (refer 4.1.1).
4. Road id in between the source (first) and destination (last) element in a chromosome (array) is randomly generated road IDs.
5. A chromosome consists of 15 elements/road IDs.
6. Population size remains throughout process.
7. Fitness Value of each chromosome is sorted. E.g. N1…..N100 fitness value.

8. **Overlapping Rate** : 60%

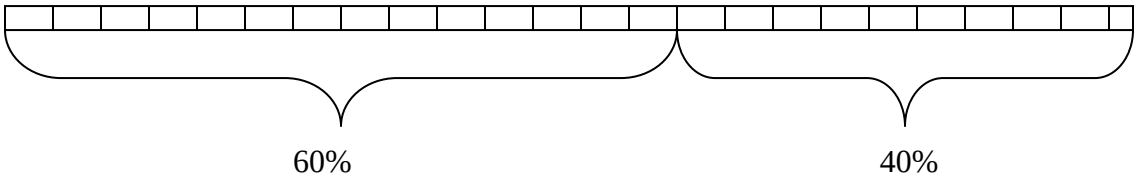

Figure 3.8 : Generation Sample of 100 population

- First 60 chromosomes from 100 chromosomes goes for Crossover, N1 – N60
- Last 40 chromosomes goes for Mutation, N61 – N100

These implies that we are letting the better chromosomes to perform cross over operations, while the bad chromosomes to perform mutation.

9. **Crossover Rate** : 70%

- From 60 chromosomes taken, randomly choose two points, N3 and N10 to cross over**.**
- Two points in the chromosomes is randomly chosen.

- Randomly generate ***a*** number, real number from 0 to 1, if the random generated number < 0.7 system will do cross over else the old chromosome will remain and brought into next generation. Therefore, the chances of it being crossed over is 70%.
- Null Value, -1 is also eligible for cross over.
- ***2*** **-** Fix Point Cross Over is used
- If Fitness Value (refer 3.3) for chromosome C1 is 23 after Cross Over compared to old value, 20 then the old chromosome is remained.
- Calculate Fitness Value of current chromosome, Do cross over, Check new Fitness Value against Old Fitness Value. If old is lesser, remain old.
- N1 = Chromosome 1

  N2 = Chromosome 2

  N1 at 3^{rd} element and 5^{th} element to be crossed over with N2 at 3^{rd} element and 5^{th} element respectively.

  After cross over $N1_{new}$ and $N2_{new}$ is produced.

All 4 chromosomes (2 old chromosomes and 2 new chromosomes) are compared and only 2 best chromosomes are brought forward to the next step.

10. Once crossover is completed, the entire 100 Chromosome is sorted based on fitness Value. The lowest fitness value (best solution path) is placed on top.

11. **Mutation** : 10%

- 40 last chromosomes are selected for Mutation.
- Chances for mutation in each element of the chromosome is 0.1
- A random number is generated each time an element is chosen from the chromosome. If randomly generated number is < 0.1 therefore Mutation is carried out else the old element is remained in the chromosome.
- Null Value also can contribute in mutation, if it is happens to be chosen as the element for mutation**.**
- Random mutate is used for mutation process.
- Each element in the array is checked with the probability of 0.1% to mutate given a random number 0 to 1.
- Randomly choose a road id to replace the old road id in the chromosome element. If the random number taken at that element is ≤ 0.1
- Eliminate the duplication of road id in a chromosome first before perform the next step.
- If the new mutated chromosome fitness value is lesser than the old one, remain the old one.

11. <u>Road Shortening</u>

- After each mutation and Cross Over has completed, Road shortening is done then fitness value is calculated.
- Repeated value in an element is placed with null, -1. Null Value is replaced for the second repeated element, except the road id located at the first and last element, which contains the source and destination location.

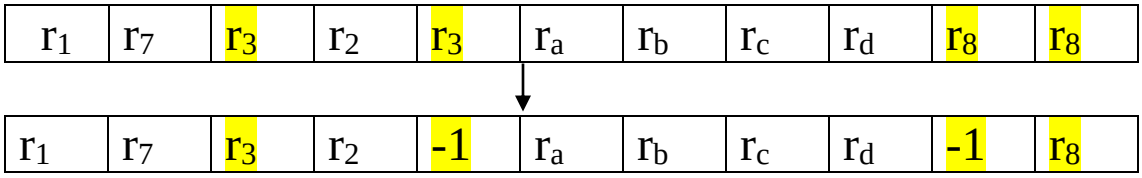

Figure 3.9: Road Shortening

12. Finding the solution path and all possible paths is based on real-time.

3.6.4 New perspective to this Simulation

The new perspective brought to this topic are :

Road Shortening

a) Check array for Duplicate road, replace the second duplicate road with null value (-1).

b) Kindly *refer section 3.6.3*

Overlapping Rate

a) Set aside the amount of population should be sent for cross-over and the balance for mutation.

b) Kindly *refer section 3.6.3*

c) If the good chromosomes are mixed with bad ones, it wastes processing time.

3.6.5 Flow Chart

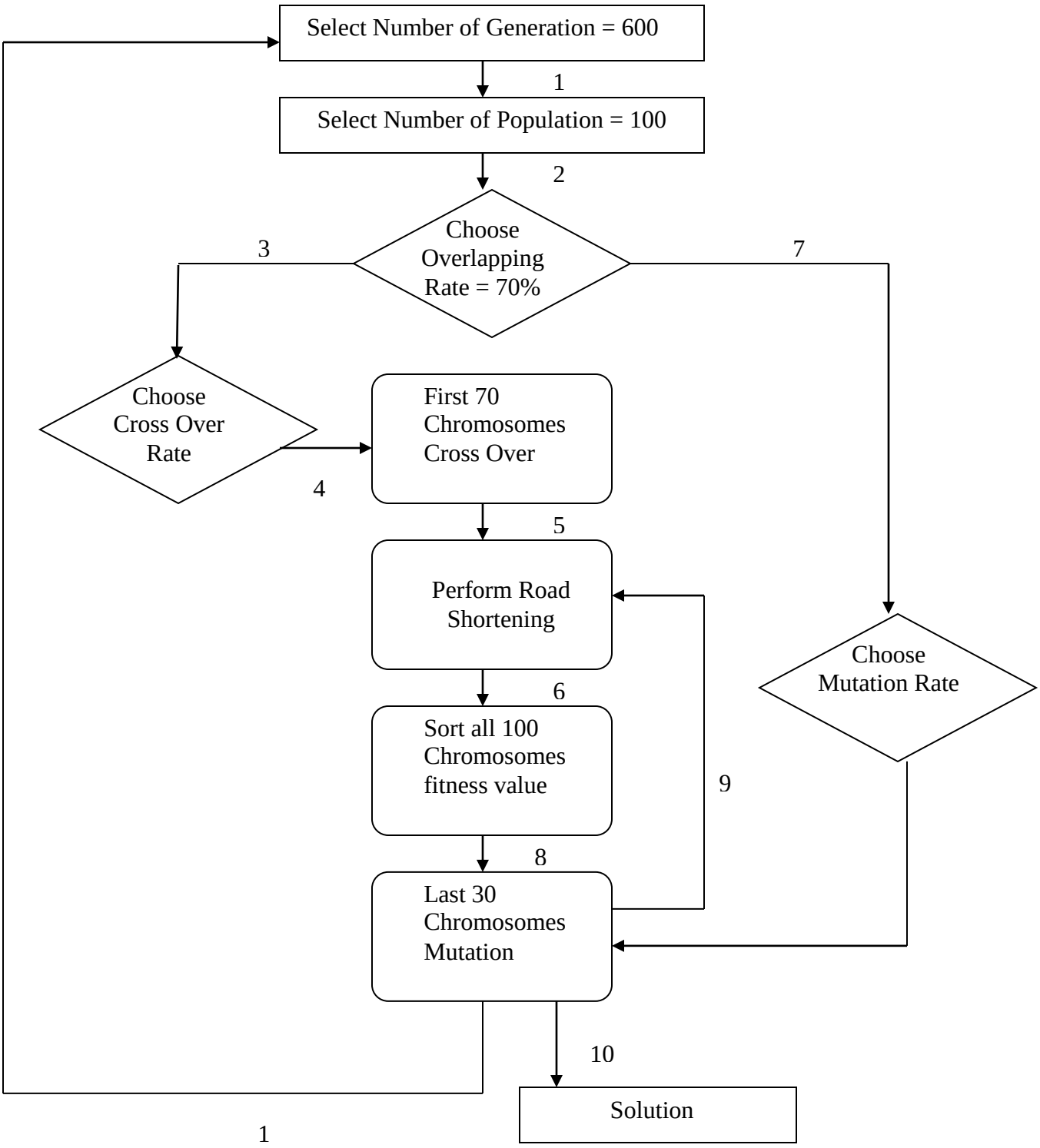

Figure 3.10 : Simulation Methodology Flowchart

3.7 Libraries/Class Representation

1. Form1.cs : Screen

2. GA.cs : GA related functions / GA components and graph

3. LinkListLib.cs : Data Structures

4. MapOperation.cs : Map, road process, routings, show/display which route being used

3.8 Software Specification

- Microsoft Windows XP (Home Edition) – Service Pack 2
- Microsoft .NET framework SDK v1.1
- Microsoft .NET Visual Studio .NET 2003 (include C#)

3.9 Hardware Specification

- Pentium 4
- Intel Centrino
- 40G Hard disk
- 504MB of RAM
- 1.73GHz Processor

3.10 Chapter Summary

This chapter illustrates the method used in this study to develop the simulation after taking into account of the literature review done on GA. How the chromosome is represented and the decision of fitness function is presented. A fixed length chromosome is used, proportional selection for selection mechanism, applied both reproduction operators which is the cross-over and mutation. Two point crossover and random mutation are applied.

Mutation is done after crossover and upon completion of the chromosome sorting. Why this routine was chosen is pointed out long with the pseudo-code and flow chart for the algorithm used for GA. Here, details on the method used for the development such as Road Shortening, Road Not Connected Weight and Pair Road Connection Weight which are a new perspective used for this development.

Chapter 4

Simulation System Design

4.1 Simulation Screen Layout

4.1.1 Vehicle Routing Properties

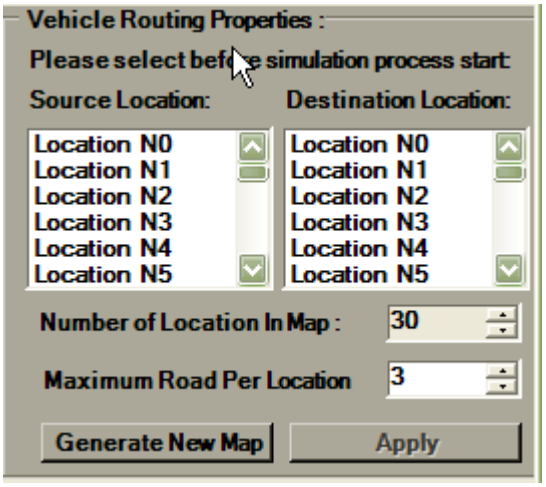

Figure 4.1 : Simulation – Vehicle Routing Properties

1. A solution should have a Source and a Destination
2. We can choose the number of nodes that should appear in a map through "Number of Location in Map"
3. Number of road connected to each one node can be determined through "Maximum Road Per Location".

Figure 4.2 : Road Connection

Figure 4.2 shows that Node N_{10} has 2 road connected to it while Node N_{15} has 3 road connected to it.

4. Nodes for each new map is randomly generated based on the "Number of Location In Map" and "Maximum Road Per Location" values set on the simulation.

5. We can choose a map of our choice by clicking the "Generate New Map" button.

6. Choosing the Source Location and Destination Location would enable the "Apply" button.

4.1.2 Solution Monitoring Process

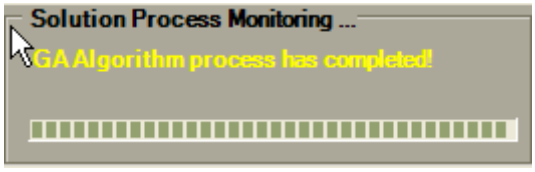

Figure 4.3 : Solution Process Monitoring – Completed

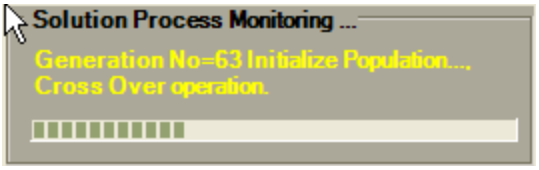

Figure 4.4 : Solution Process Monitoring – Crossover Operation

1. The simulation progress can be viewed at this caption.
2. It would display
    - Population Initialization
    - Generation Number
    - Operation it is currently working on e.g. Crossover or Mutation
    - Process has completed

4.1.3 Map Screen

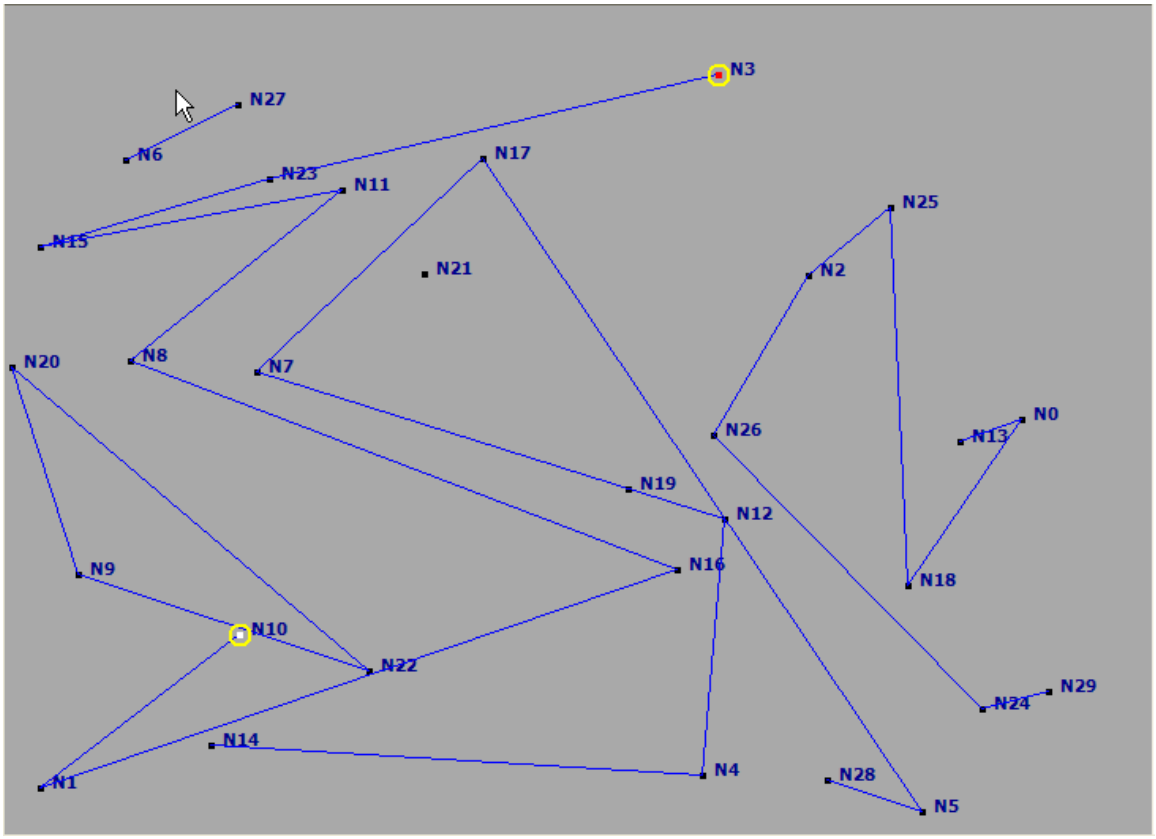

Figure 4.5 : Initial Map with Source and Destination Selected

1. Once the Source and Destination is selected in the Vehicle Routing Properties (refer section 4.1.1) and the "Apply" button is selected the map will display as in Figure 4.1
2. Red circle as the Source and the White circle as the Destination.

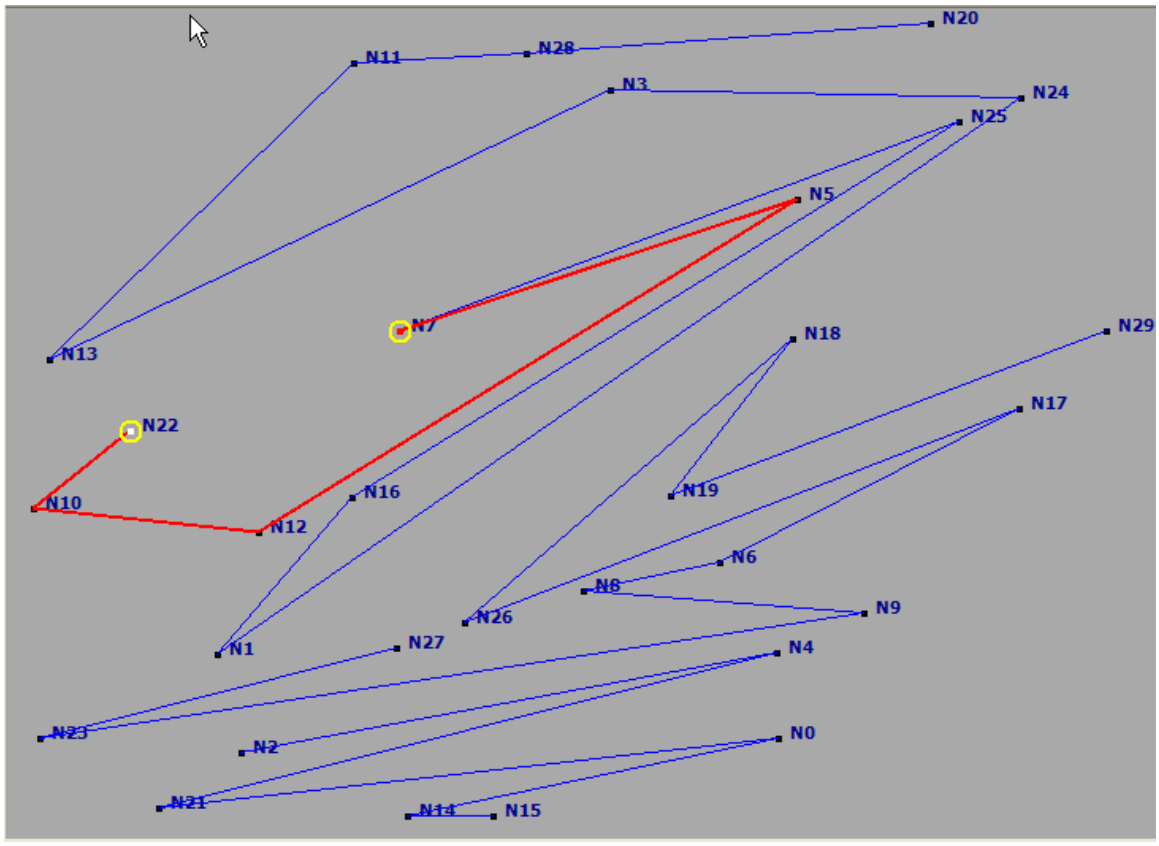

Figure 4.6 : Map with best route display

3. Once the simulation is completed, the best route or shortest path is displayed in red with the source connected to the destination.

4.1.4 Best Path Solving Process Information

1. Best Path Solving Process would display
    - Algorithm Used e.g. Djikstra or Genetic Algorithm or Full Search
    - If the Source Road and Destination Road is connected before processing
    - The best path route
    - Time taken for this best path from to travel from Source to Destination
    - Distance traveled from Source to Destination

- Computational/Simulation Time

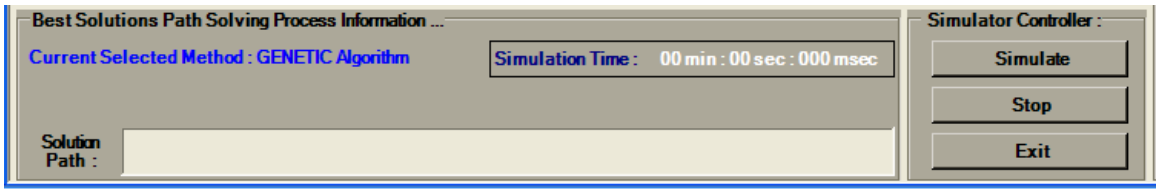

Figure 4.7 : Solution Path – Initial

The above is the standard state of the Solution Path Interface before simulation

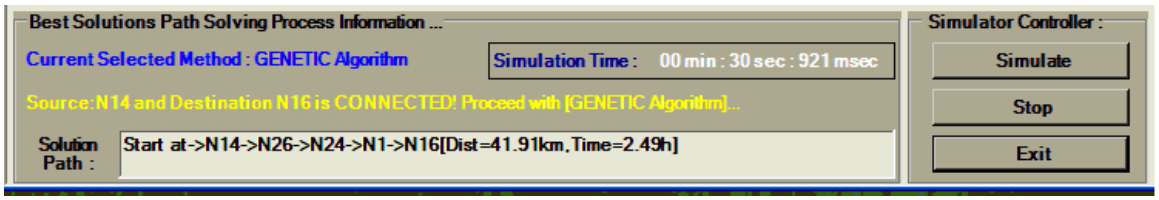

Figure 4.8 : Solution Path - Source and Destination Connected

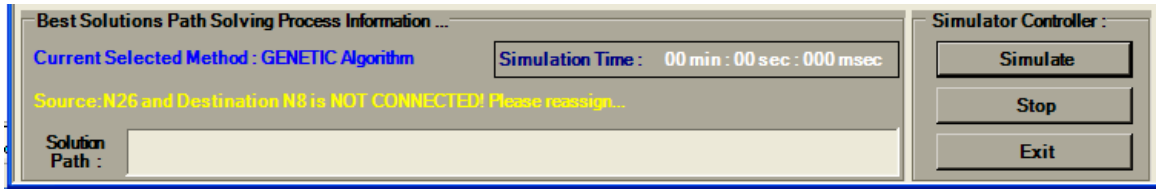

Figure 4.9 : Solution Path – Source and Destination Not Connected

Once the Source and Destination is chosen from the Vehicle Routing Properties (refer Figure 4.1) and the Simulation button is started, if the Source and Destination are connected than the simulation would proceed and the status would display that the "Source and Destination is CONNECTED" as in Figure 4.8 else if the Source and Destination is not connected than the Figure 4.9 would be displayed.

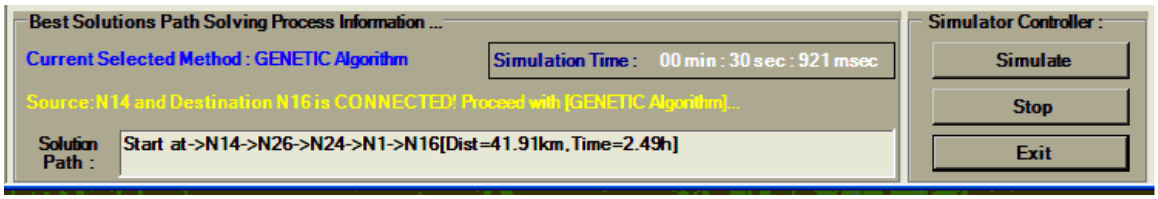

Figure 4.10 : Solution Path – Process Completed

Once the simulation is completed, Best Path, Distance Traveled, Travel Time Taken and Computation Time will be displayed as in Figure 4.10

4.1.5 Main Function

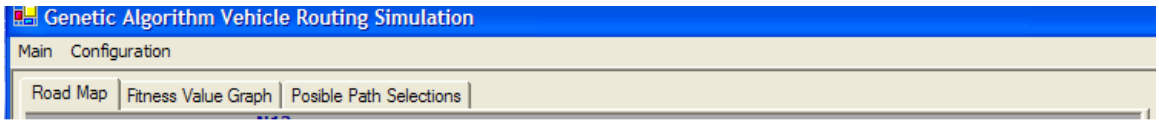

Figure 4.11 : Main Function Header

1. [Main] consists “Exit” to exit the program.
2. [Configuration] will display the algorithms that can be used eg.GA, Djikstra and Full Search.
3. [Road Map] displays the Map with roads and nodes.
4. [Fitness Value Graph] displays graph of the Fitness Value against the Generation. This Graph is only available for Genetic Algorithm.

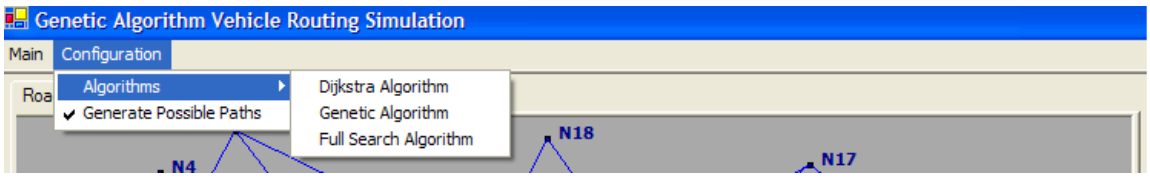

Figure 4.12 : Main Function Configuration

5. To choose from Dijkstra or Genetic Algorithm or Full Search.

4.1.6 Road Condition Setting

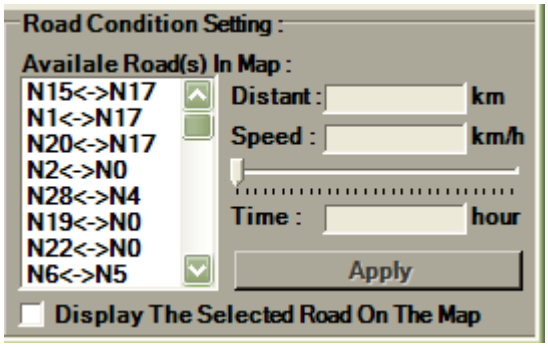

Figure 4.13 : Road Condition – Initial

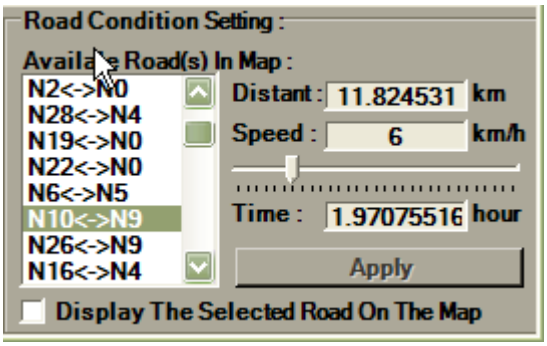

Figure 4.14 : Road Condition – Selected

1. Distance is displayed in kilometers. Road distance is a constant value.
2. Time is displayed in hours.
3. Speed is calculated from Distance and time by the formula :

$$\text{SPEED} = \frac{DISTANCE}{TIME}$$

Therefore, Time can be calculated as :

$$\text{TIME} = \frac{DISTANCE}{SPEED}$$

4. Assumption on Road condition (eg. Traffic congestion) can be made where we can adjust the speed to travel the designated road, so based on the formula above, time taken to travel can be calculated.
5. Therefore, speed on each road can be assigned manually as desired.
6. Time taken to travel the road can be changed and the "Apply" button is enabled.
7. To display the road selected click the check box "Display Selected Road on the Map"
8. For example, lets say Source is N3 and Destination is N23, but we would like to set the road condition for Road N12 – N20, we can change the speed and display that road on the map as shown in Figure 4.15

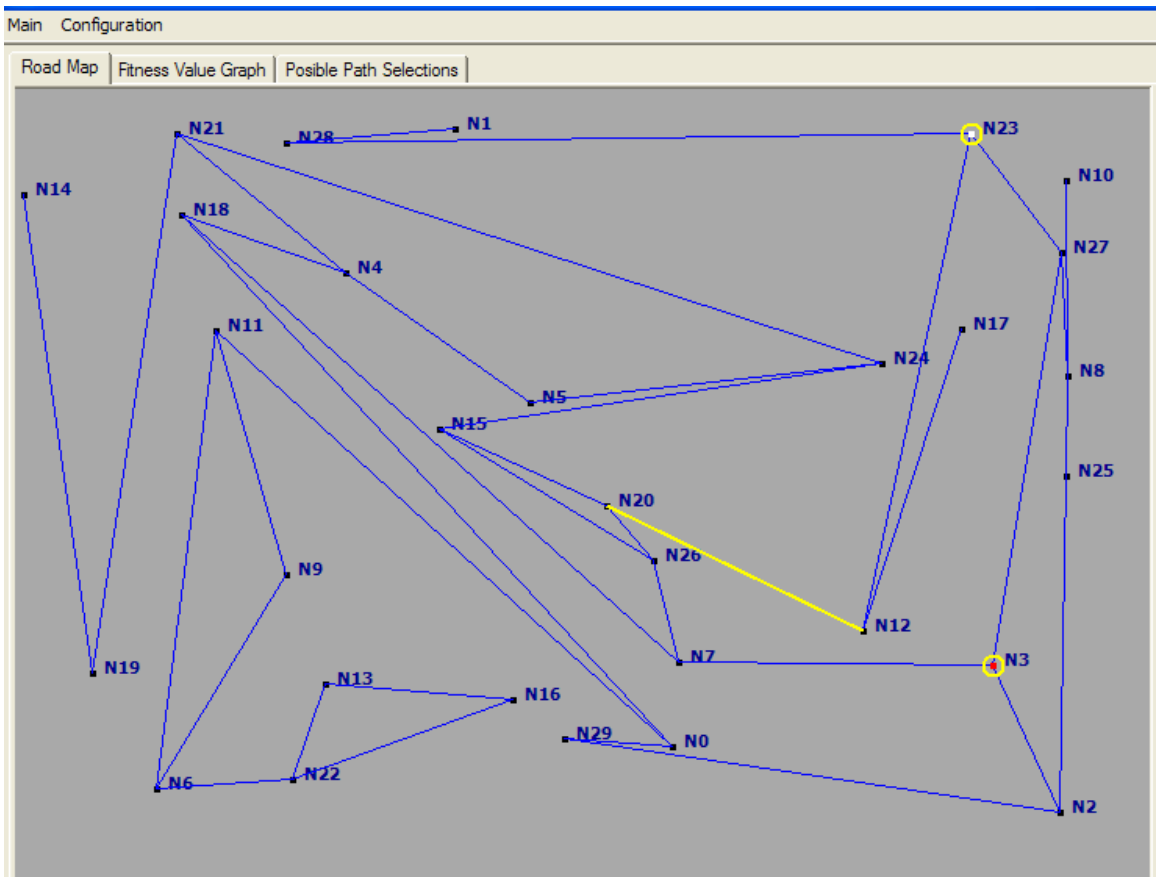

Figure 4.15 : Road Condition Setting – Display on Map

4.1.7 Fitness Graph Screen

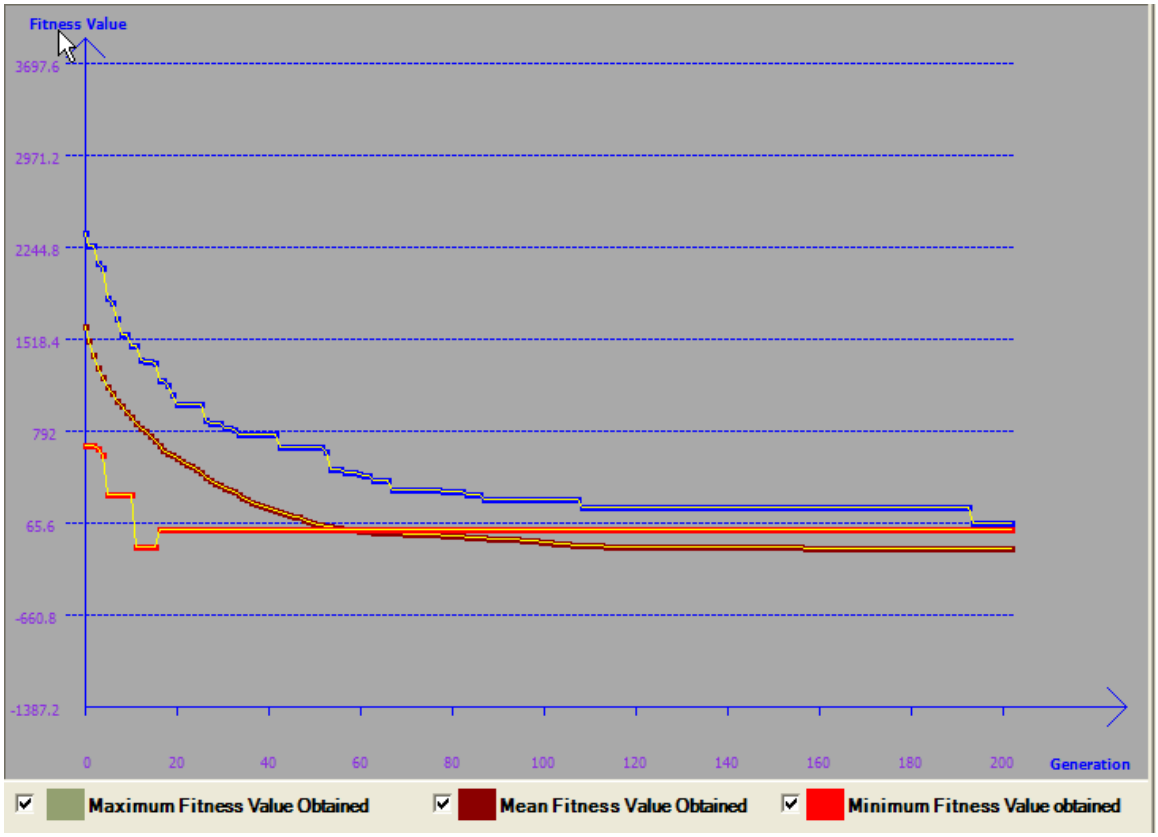

Figure 4.16 : GA Fitness Value for Generation Graph

1. Y axis : Fitness Value (increased downwards), in this simulation; the lower the value of fitness, gives a better solution.
2. X axis : Number of Generation
3. Blue : Maximum Fitness Value Obtained
4. Maroon : Mean Fitness Value Obtained
5. Red : Minimum Fitness Value Obtained

$$\text{Maximum Fitness : } \frac{\sum_{i=1}^{900} c1+i}{900} \quad \text{where 900 = Population size}$$

6. Graphic User Interface (GUI) is developed using GDI (Graphical Development Interface) in the Microsoft Graph representation.

4.1.8 Full Search

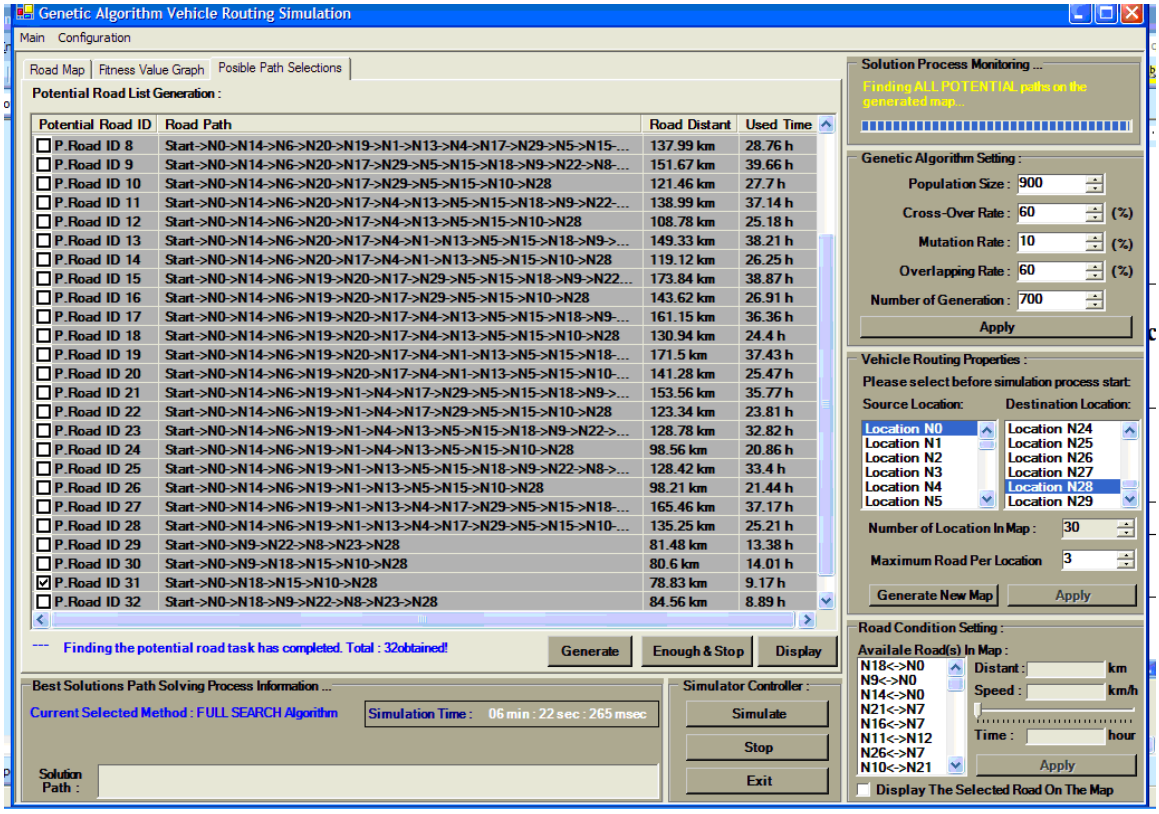

Figure 4.17 : Full Search/Possible Path Screen

1. This is the Full Search algorithm, where it would display all possible path.

2. [Generate] button would start to compute all possible path.

3. [Enough & Stop] allows us to stop the computation.

4. Upon completion of getting all possible path, system would automatically ‘tick’ the check box with the shortest path is distance. User could use that checked solution or any other solutions provided in the list.

5. [Display] would take us back to the map, where the shortest path found by Full Search is displayed.

4.1.9 Genetic Algorithm Settings

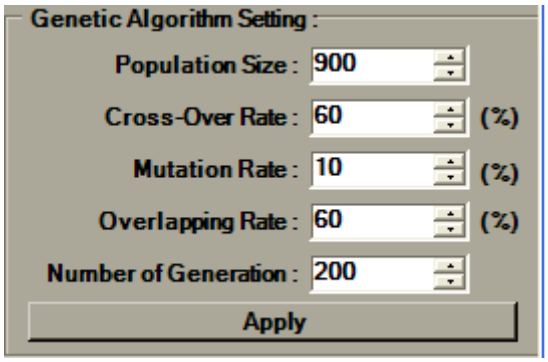

Figure 4.18 : Genetic Algorithm Settings

The above would enable us to set the GA settings we would like to test. Simulation would process till the number of generation is reached or rather the stop criteria is met.

4.1.10 Djikstra Algorithm

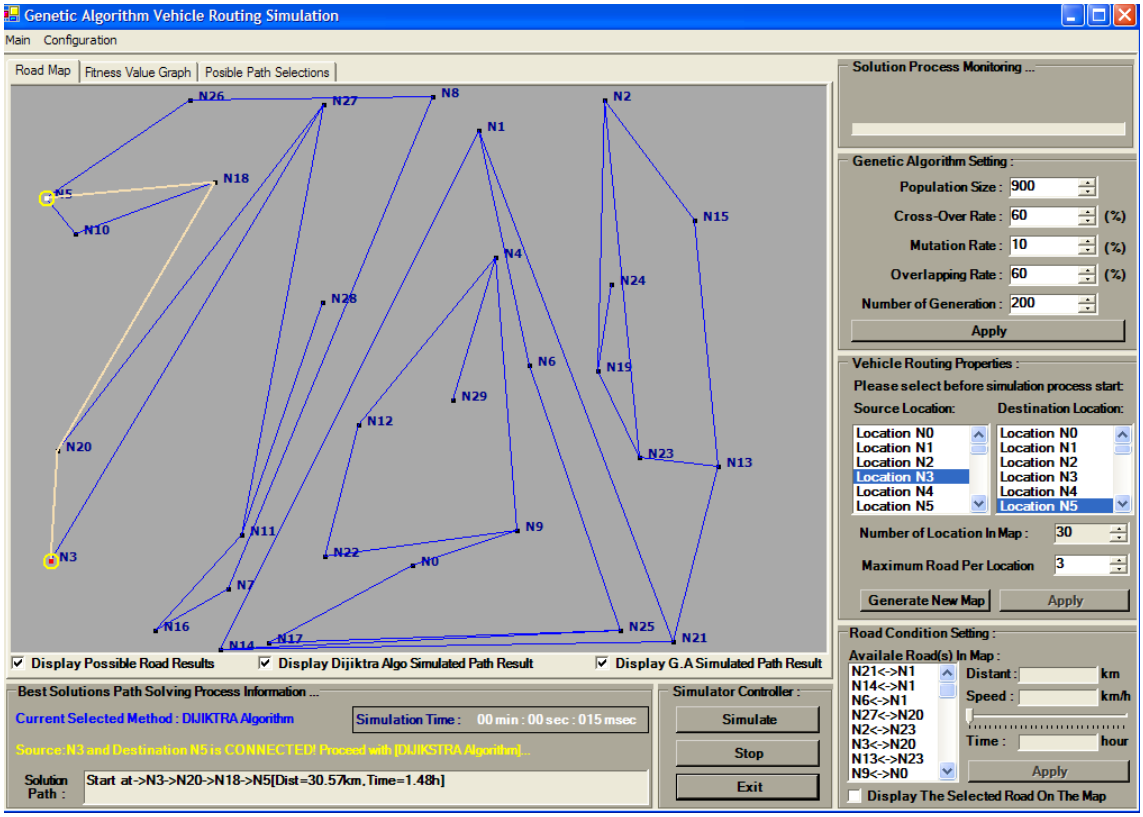

Figure 4.19 : Djikstra Algorithm Solution Display

Djikstra solution path is displayed in beige colour.

Simulation Time and Solution Path is displayed. Details on Djikstra Algorithm kindly refer section 2.10

4.1.11 Display of all three Algorithms

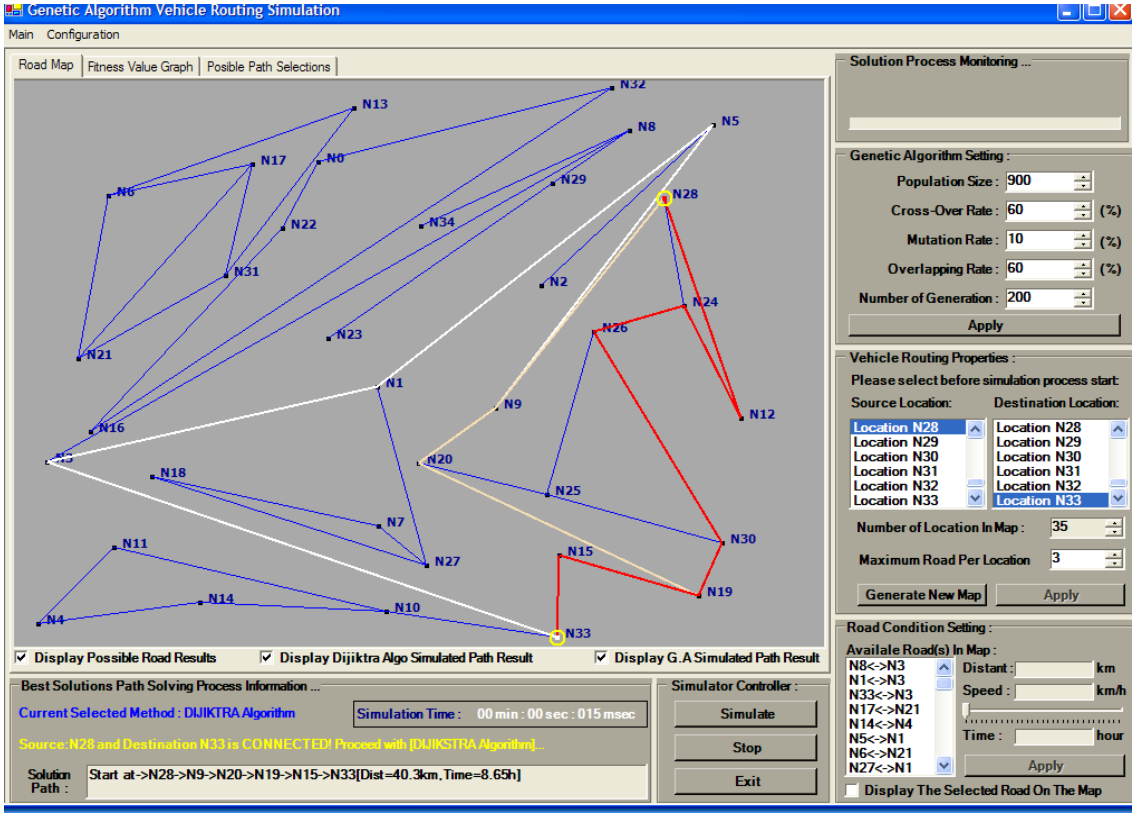

Figure 4.20 : Display of 3 Algorithm on SHAVRS Map

White : Full Search

Beige : Djikstra

Red : Genetic Algorithm

Note : The above simulation is not to analyze the difference between the three algorithm.

This is merely to display all the three algorithm on the map.

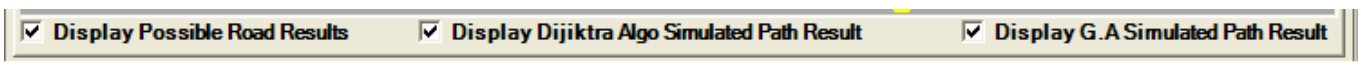

Figure 4.21 : Three Algorithm Display Bar

This bar would enable us to choose which algorithm we would like to put on view in the map.

4.2 Chapter Summary

This chapter takes you through the SHAVRS system design. You will get to virtually vision the look of the simulation and the development life cycle. Along the phase of development, the system was scrutinized by author's supervisor; enhancement and new features had been added based on feedbacks and ideas.

Chapter 5

Results and Analysis

The section 5.1 would highlight the assumptions made for the SHAVRP simulation. Moving on the section 5.2, 5.3, 5.4, 5.5 and section 5.6 takes the reader through the testing that had been carried out. The test result for each of the above section is revealed in section 5.6, while the analysis on the results attained is rationalized and explained in section 5.7 and section 5.8.

5.1 Assumptions

- No intersections/crossroads between roads/routes/path
- No direction (not vector/ not a directional map)
- Condition of a road is the same

  e.g. N12 to N7 or N7 to N12 are the same condition

5.2 Testing on the Genetic Algorithm Parameters

5.2.1 Population Size and Number of Generation as Constants

Population Size : 1000

No. of Generation : 200

Source : N18 Destination : N16

Possible Solution :

a) N18 – N13 – N21 – N7 – N16

b) N18 – N13 – N21 – N29 – N26 – N7 – N16

c) N18 – N13 – N21 – N29 – N26 – N7 – N28 – N16

A) Overlapping Rate and Mutation Remains Same

Table 5.1 : Influence on Cross-over Rate

Overlapping Rate	50	50	50
Crossover Rate	80	60	40
Mutation Rate	10	10	10
Results	N18-N13-N21-N26-N7-N28-N16 [Dist=64.49km,Time=5.09h]	N18-N13-N21-N16 [Dist=48.74km,Time=3.47h]	N18-N13-N21-N16 [Dist=48.74km,Time=3.47h]

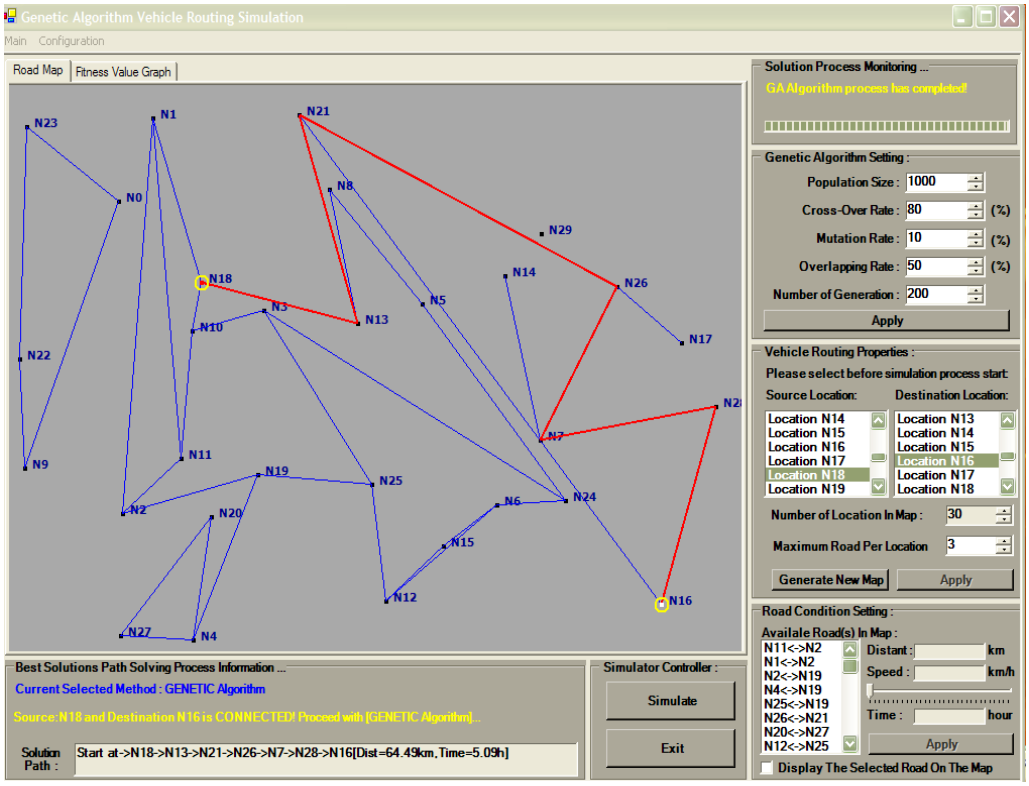

Figure 5.1 : Crossover Rate 80%

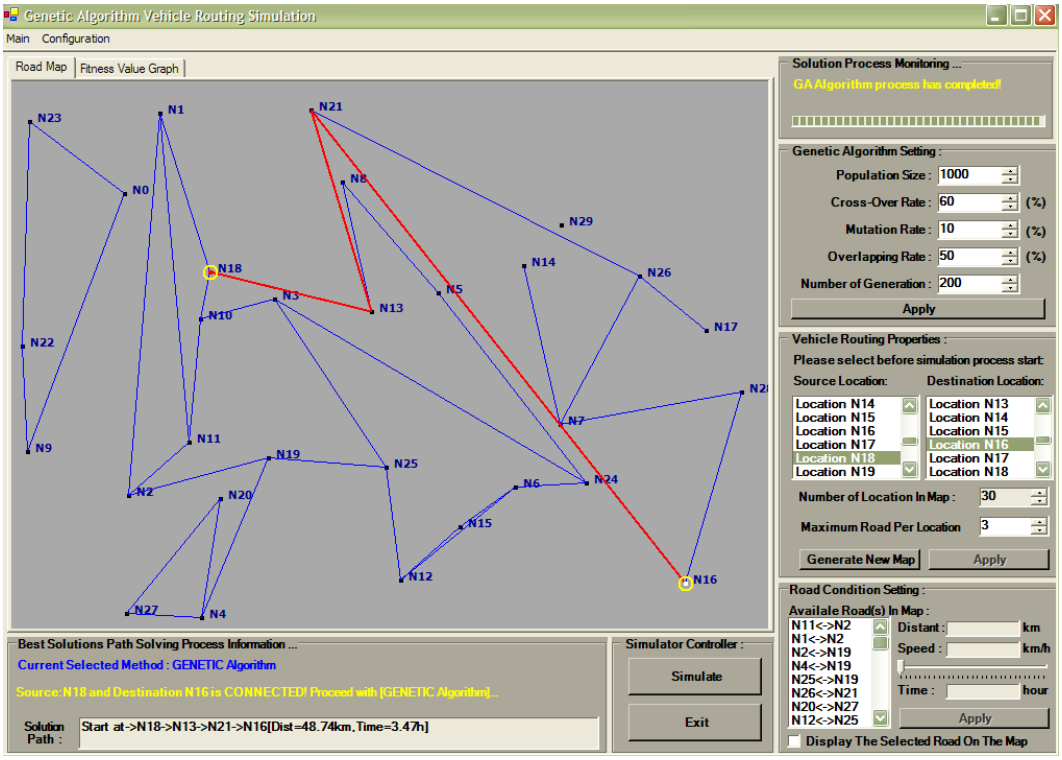

Figure 5.2 : Crossover Rate 60%

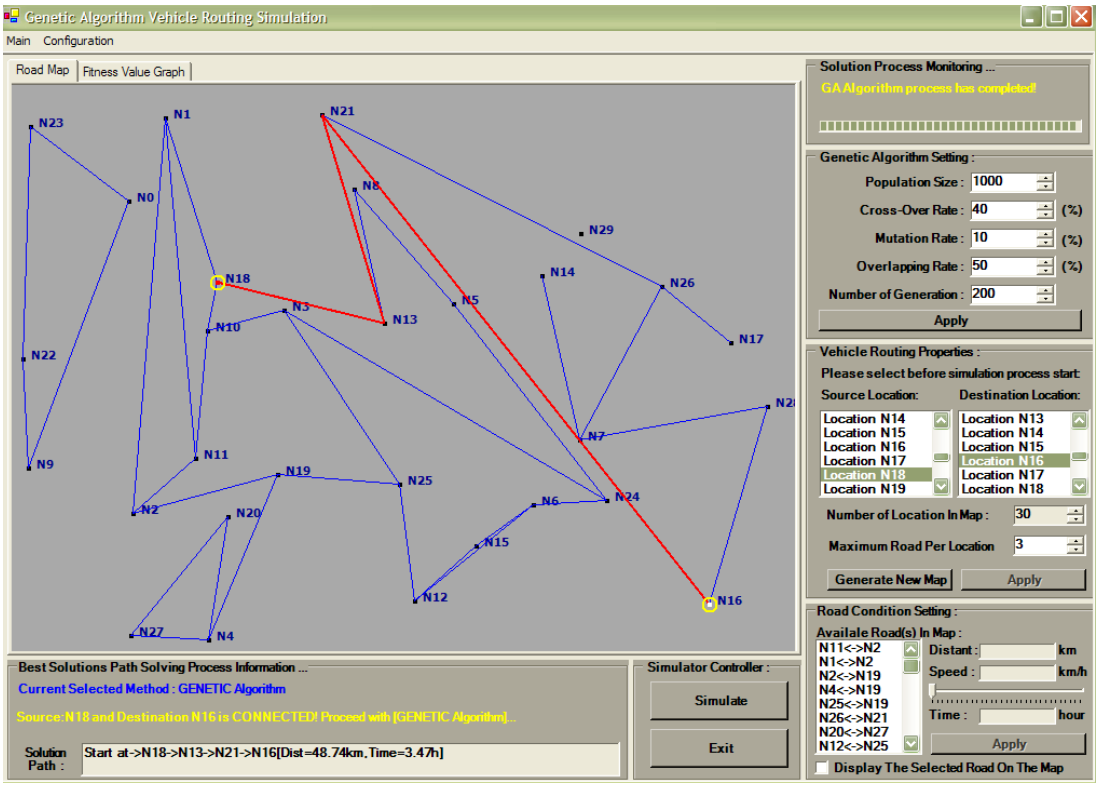

Figure 5.3 : Crossover Rate 40%

B) Overlapping Rate and Cross over remains same

Table 5.2 : Influence on Mutation Rate

Overlapping Rate	50	50	50
Crossover Rate	60	60	60
Mutation Rate	10	20	50
Results	N18-N13-N21-N26-N7-N28-N16 [Dist=64.49km,Time=5.09h]	N18-N13-N21-N16 [Dist=48.74km,Time=3.47h]	Can't Get Correct Solution. Perform more generation to obtain solution path!

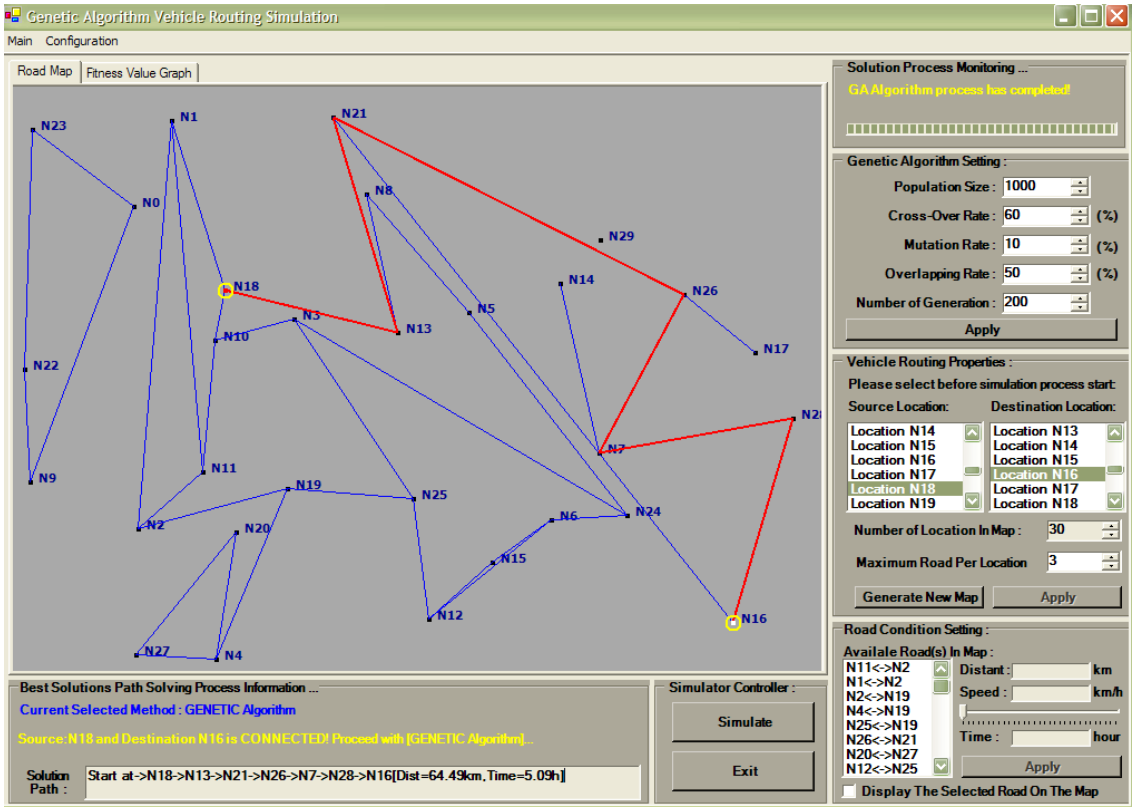

Figure 5.4 : Mutation Rate 10%

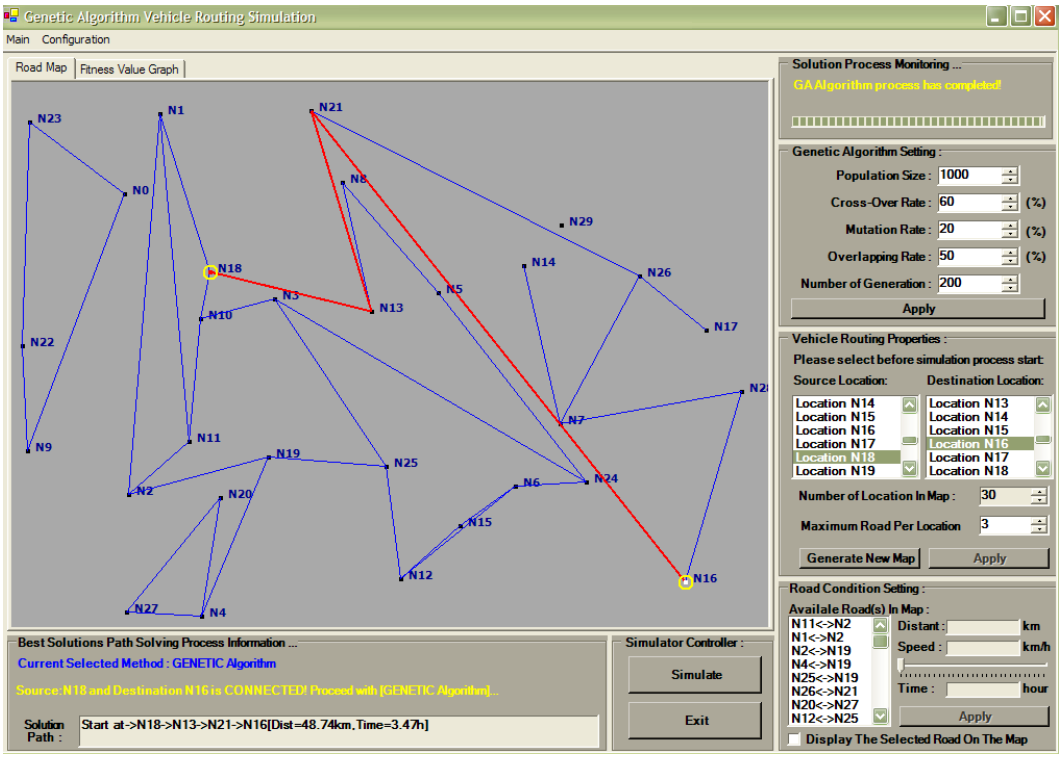

Figure 5.5 : Mutation Rate 20%

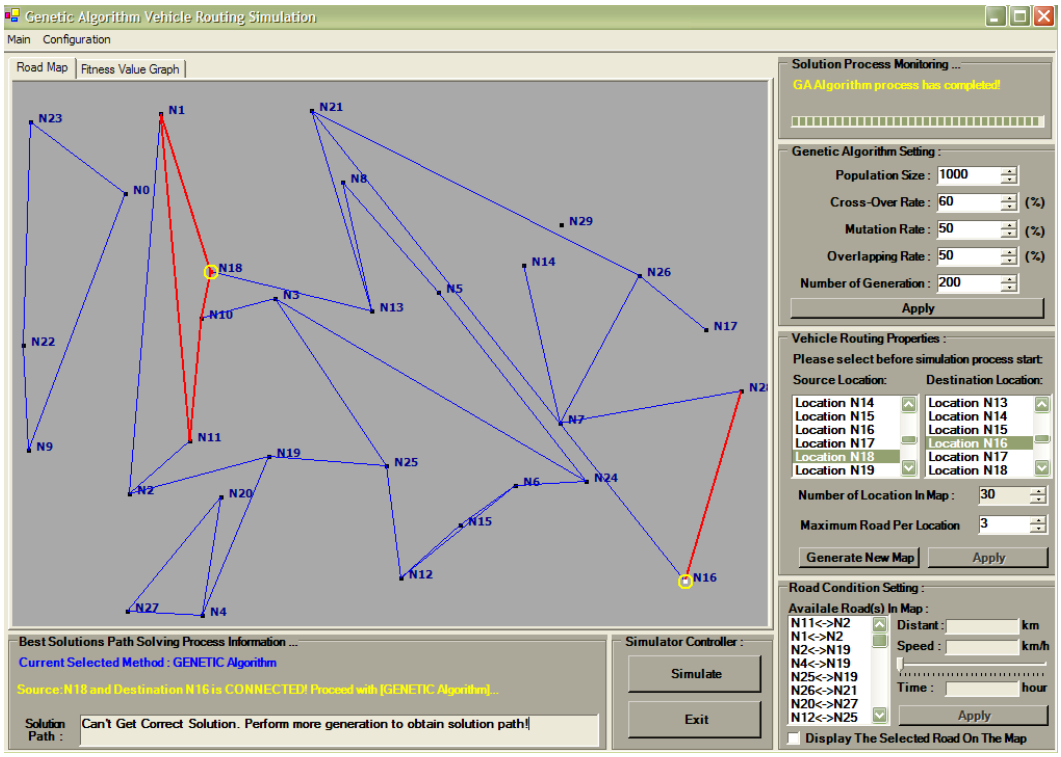

Figure 5.6 : Mutation Rate 50%

5.2.2 Number of Generation as Variable

Population Size : 1000

Overlapping Rate : 60%

Source : N8 Destination : N19

Possible Solution :

a) N8 – N6 – N7 – N22 – N19

b) N8 – N17 – N19

c) N8 – N17 – N24 – N27 – N18 – N19

d) N8 – N29 – N24 – N17 – N19

e) N8 – N29 – N24 – N27 – N18 – N19

Table 5.3 : Influence on Number of Generation

Crossover Rate	60	60	60
Mutation Rate	10	10	10
No Of Generation	200	300	500
Results	N8-N17-N24-N27-N18-N19 [Dist=70.53km, Time=7.87h]	N8-N6-N7-N22-N0-N14-N15-N18-N19 [Dist=101.53km, Time=17.14h]	N8-N6-N7-N22-N19 [Dist=54.01km, Time=6.58h]

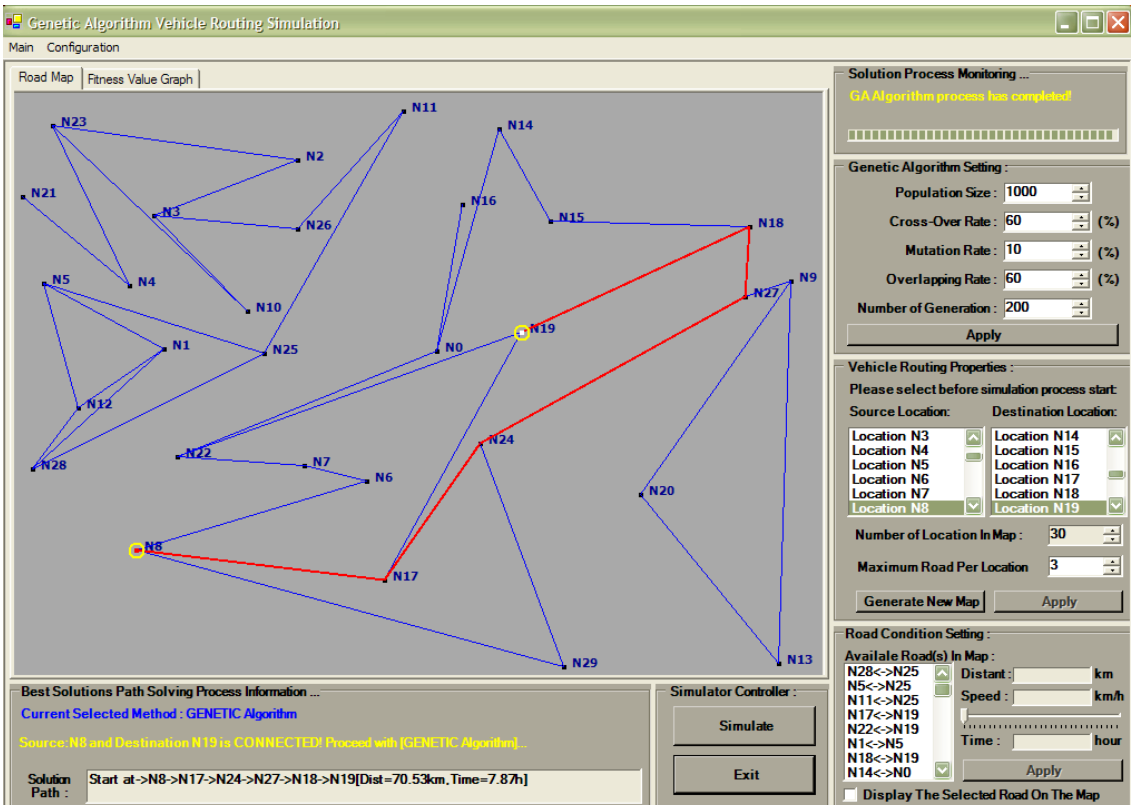

Figure 5.7 : Number of Generation 200

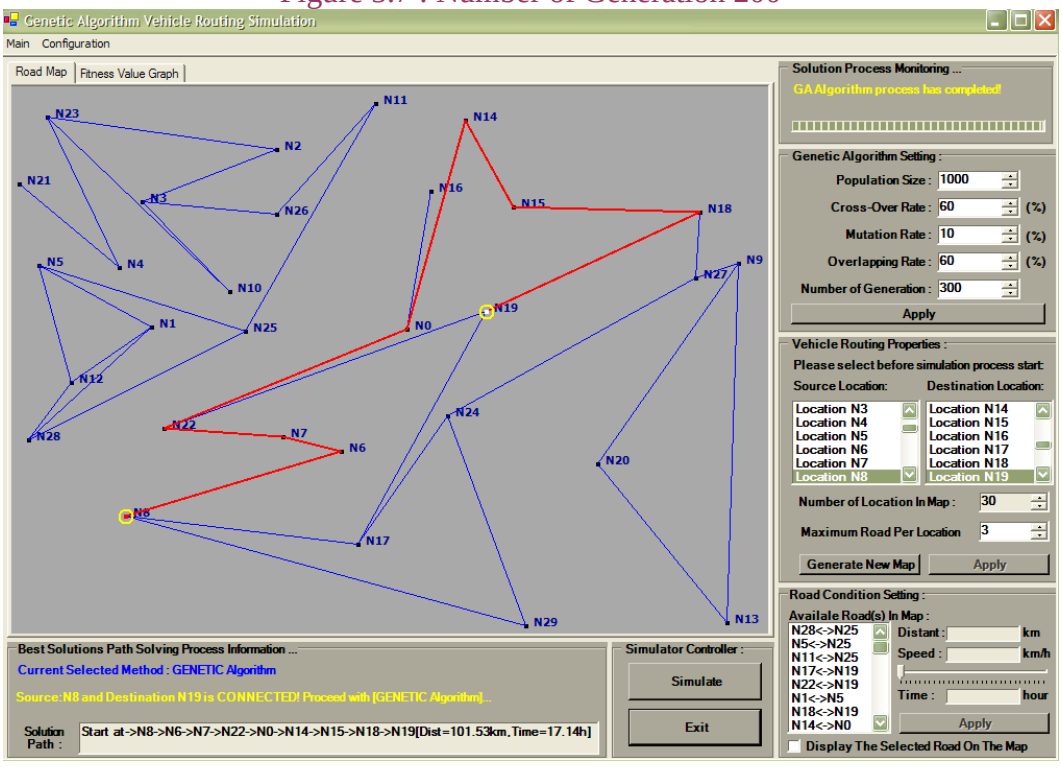

Figure 5.8 : Number of Generation 300

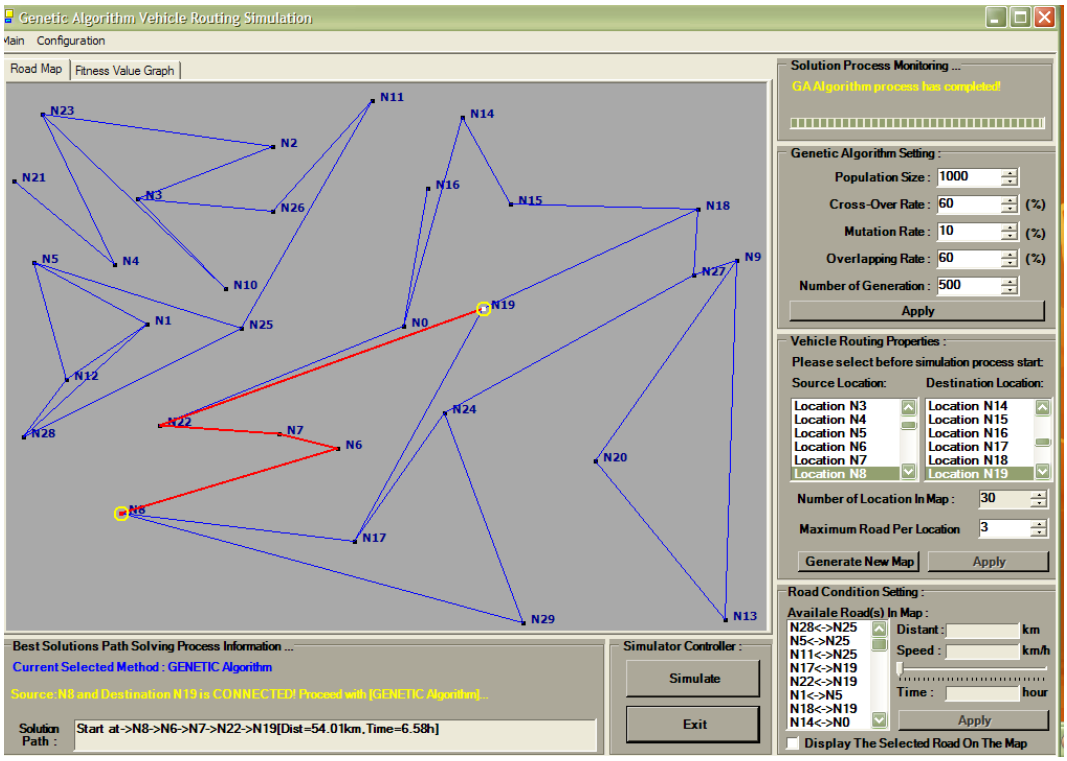

Figure 5.9 : Number of Generation 500

5.3 GA Computational Time vs Djikstra

The figures 5.10, 5.11, and 5.12 would display the computation time for GA with different number of generation while figure 5.13 would display the computation time for Djikstra algorithm. The summary of each computation time is exhibited in the table 5.4 and table 5.5 respectively. This computational time is based on 30 nodes. Table 5.12 would show the results on the increment of number of nodes for the Djikstra computation time.

Using GA

Population : 300

Time taken : 42.546sec

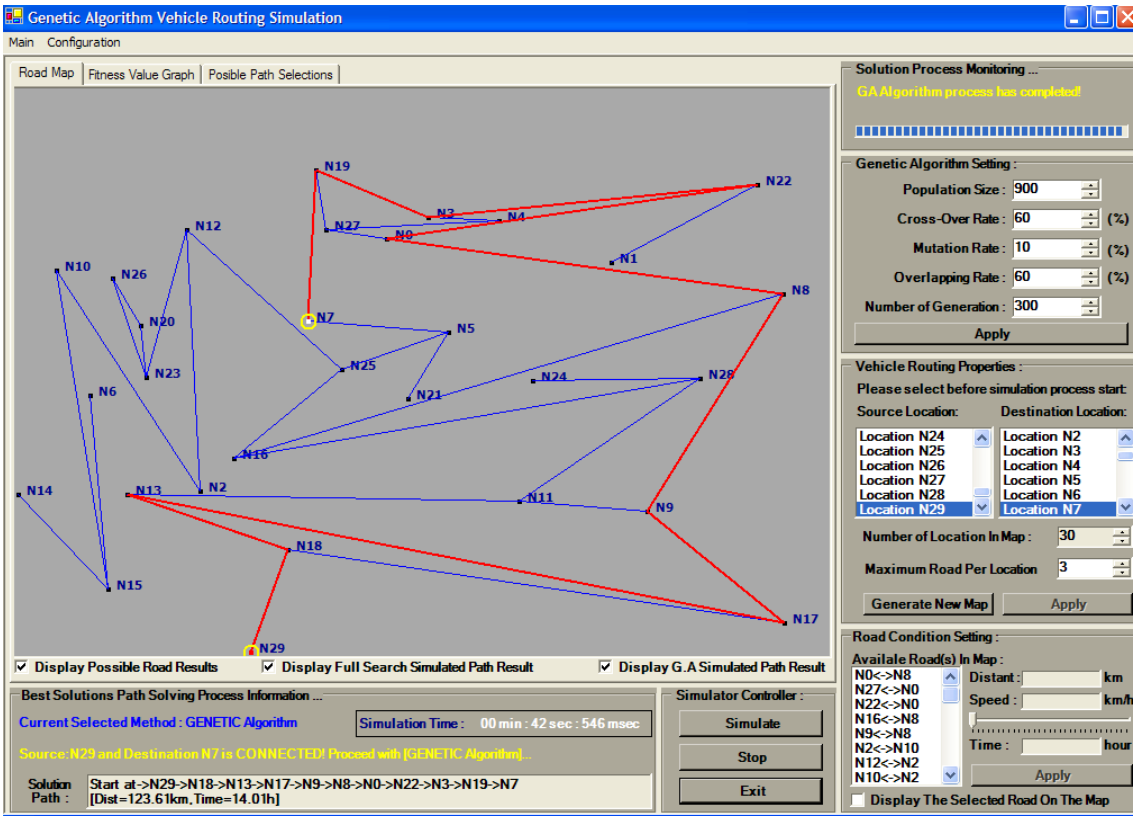

Figure 5.10 : GA Computational Time for 300 Generation

Using GA

Population : 500

Time taken : 1min 9.109sec

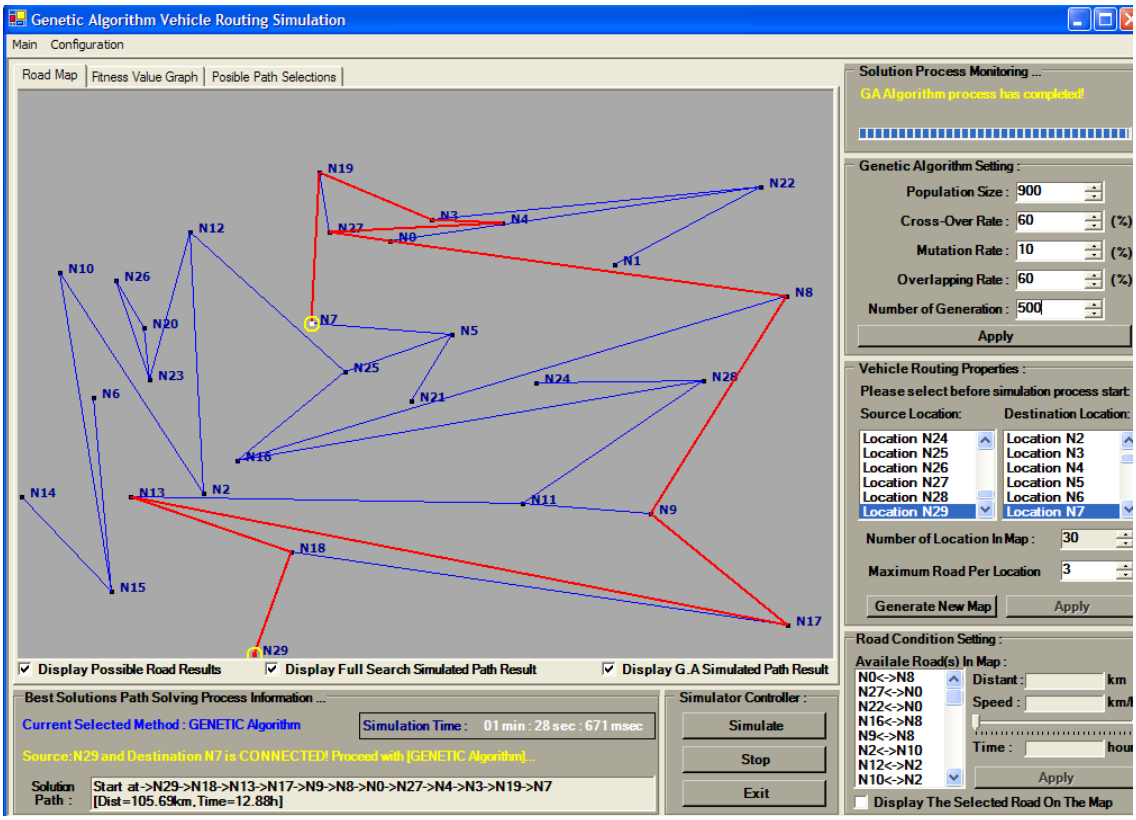

Figure 5.11 : GA Computational Time for 500 Generation

Using GA

No Generation : 700

Time taken : 1min 32.890sec

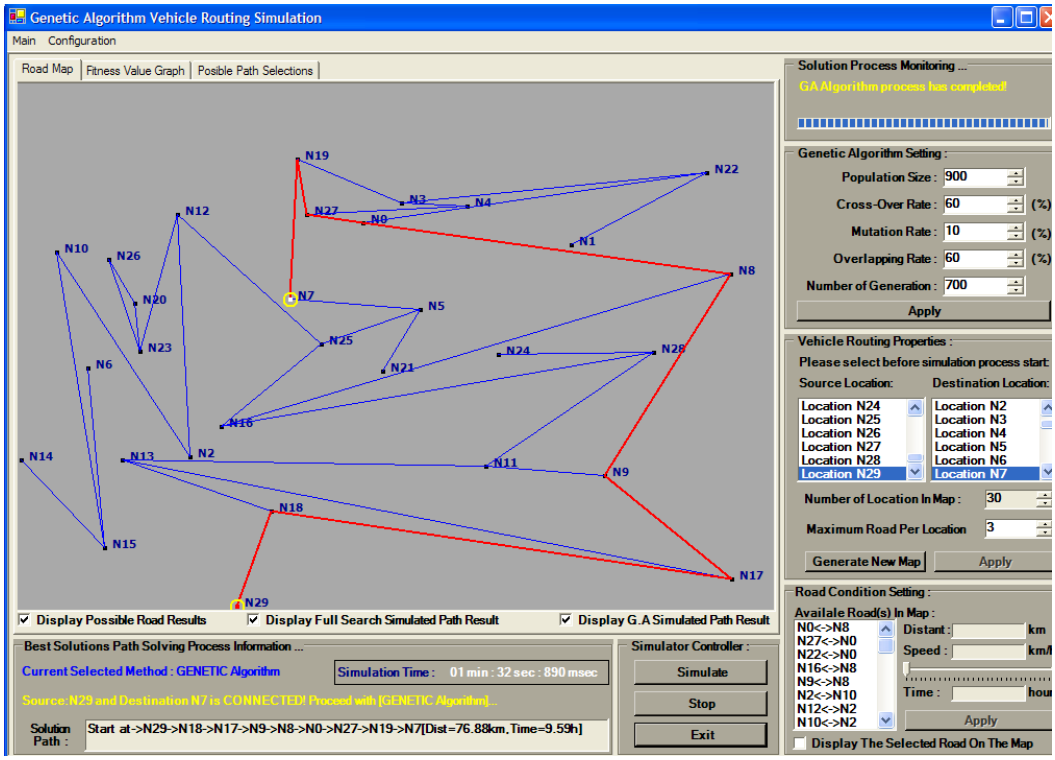

Figure 5.12 : GA Computational Time for 700 Generation

Table 5.4 : Computational time for Genetic Algorithm

No of Gen	300	500	700
Computational Time (seconds)	42.546	69.109 (1min 9.109sc)	92.890 (1min 32.890sc)
Distance (km)	123.61	105.69	76.88
Travel Time (hours)	14.01	12.88	09.59

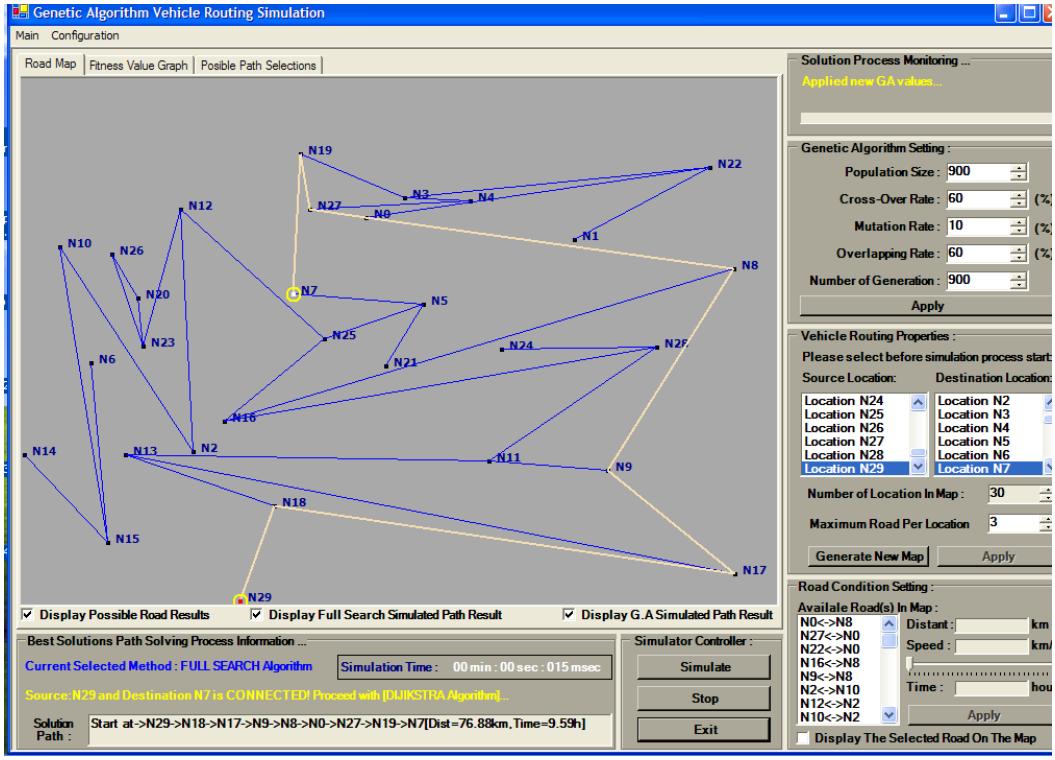

Figure 5.13 : Djikstra Computational Time

Using Djikstra

Time taken : 0.015sec

Table 5.5 : Computational time for Djikstra

Computational Time (seconds)	0.015
Distance (km)	76.88
Travel Time (hours)	09.59

5.4 Analysis on Fitness Value vs. Number of Generation Graph

The development of Vehicle Routing Simulation to determine the shortest path has also been fitted with the Fitness Value Graph. For this simulation, it is formulated that; the lower the Fitness Value; the better Solution Path is produced. We can observe from the graphs that the more we increase the number of generation, the better the fitness value.

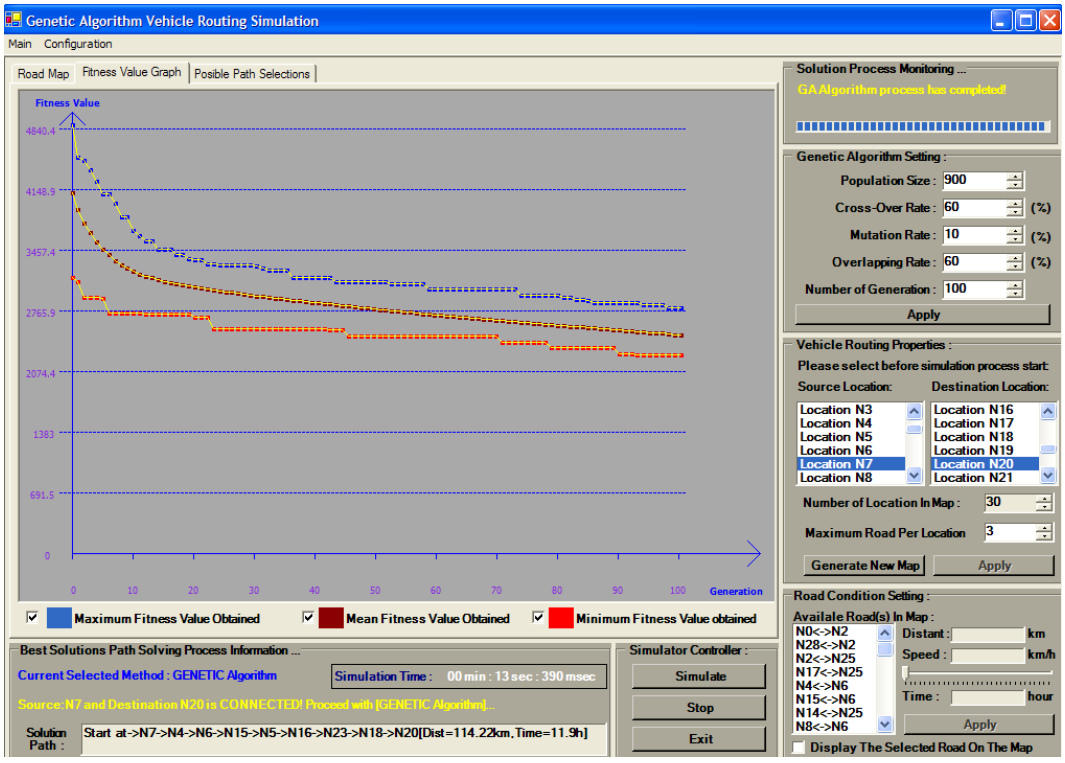

Figure 5.14 : Fitness Graph for 100 Generation

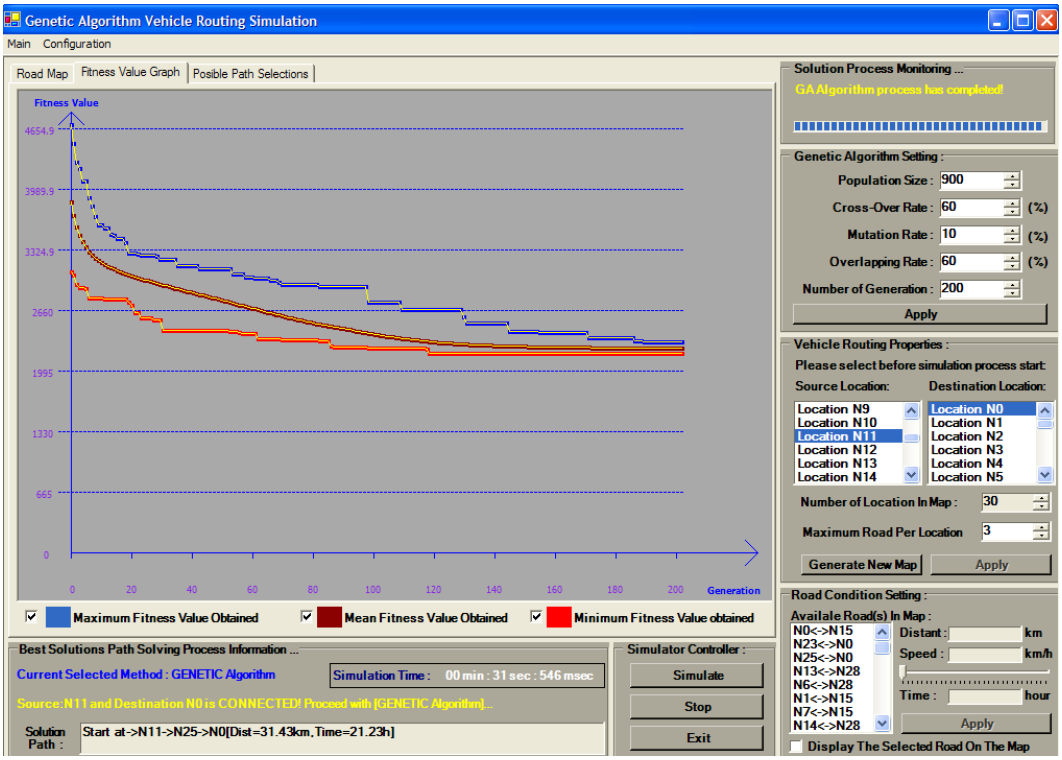

Figure 5.15 : Fitness Graph for 200 Generation

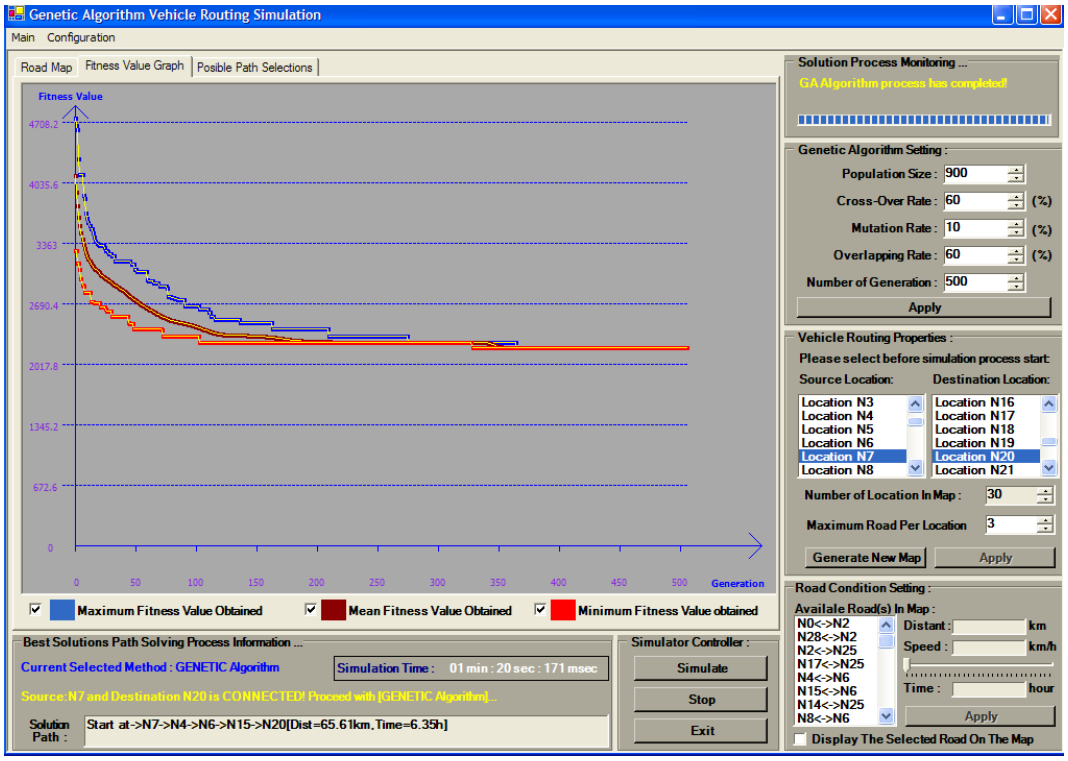

Figure 5.16 : Fitness Graph for 500 Generation

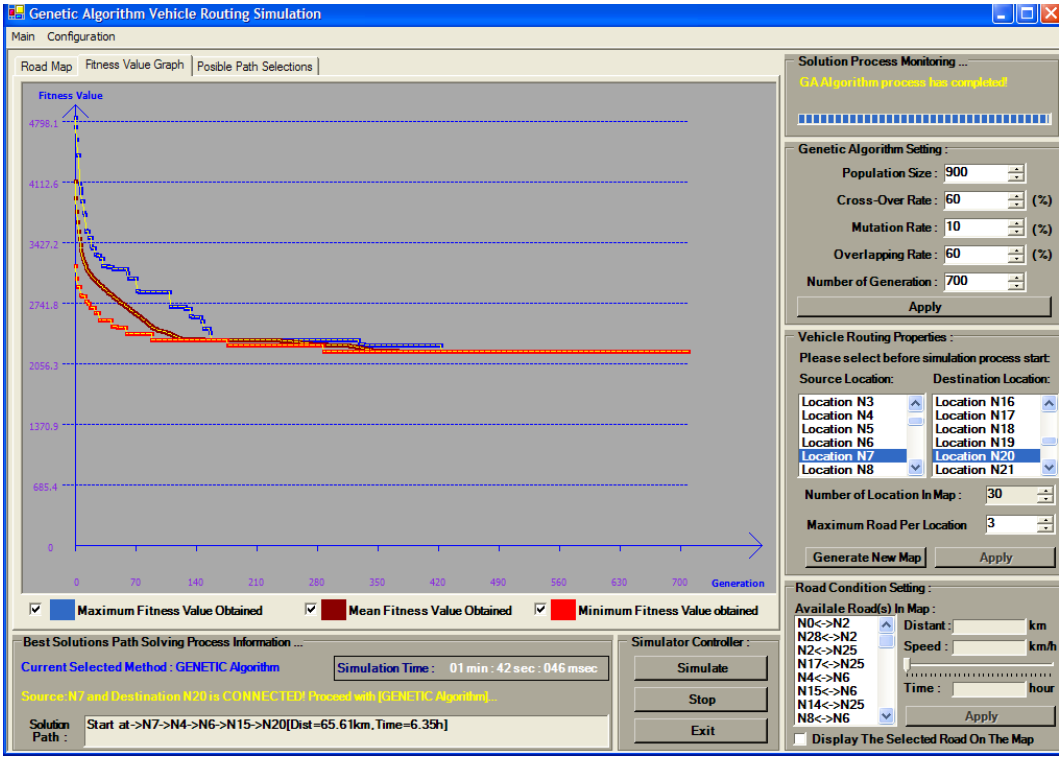

Figure 5.17 : Fitness Graph for 700 Generation

5.5 GA Computational Time vs. Full Search

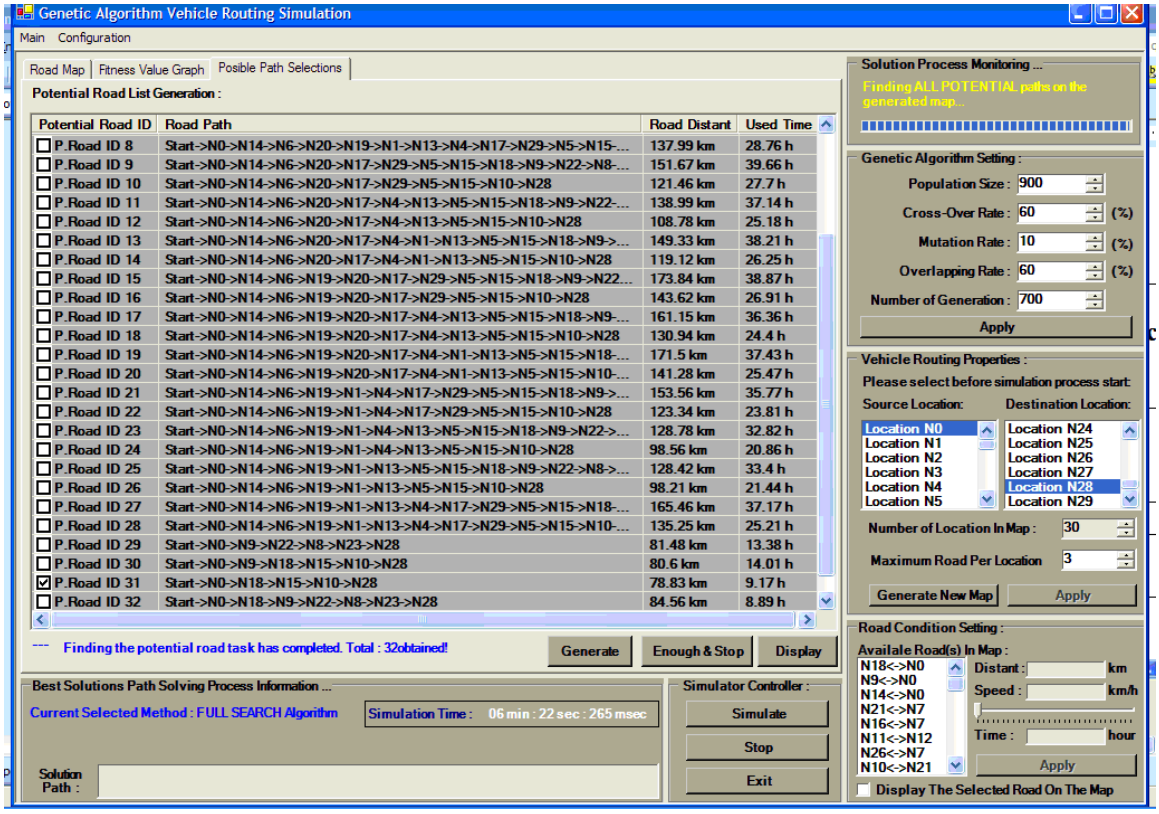

Figure 5.18 : Full Search Possible Path Computation Time

Table 5.6 : Computational for Full Search vs. GA

Algo Used / Results	GA 300 Generation	GA 500 Generation	GA 700 Generation	Full Search
Computational Time (seconds)	42.531	72.796 (1min 12.796sc)	101.781 (1min 41.781sc)	382.265 (6min 22.265sc)
Distance (km)	81.48	84.56	78.83	78.83
Travel Time (hours)	12.26	8.97	09.17	09.17

5.6 No. of Successful Search vs. Generation and Population

Number of success obtaining solution / Total number of simulation

**Set total number of simulation carried out = 5

Table 5.7 : Number of success obtaining solution / Total number of simulation

Number of generation / Number of population	100	200	300	400	500	600	700	800	900	1000
200	0/5	0/5	1/5	1/5	1/5	1/5	2/5	2/5	4/5	4/5
400	0/5	0/5	1/5	1/5	1/5	2/5	2/5	2/5	3/5	4/5
600	1/5	1/5	1/5	2/5	2/5	3/5`	3/5	3/5	4/5	4/5
800	2/5	2/5	2/5	3/5	3/5	4/5	4/5	4/5	4/5	5/5
1000	2/5	2/5	3/5	3/5	3/5	4/5	5/5	5/5	5/5	5/5

5.7 Speed against Traveling Time

5.7.1 Speed function walkthrough

This is how the screen would like before any selection, Figure 5.19

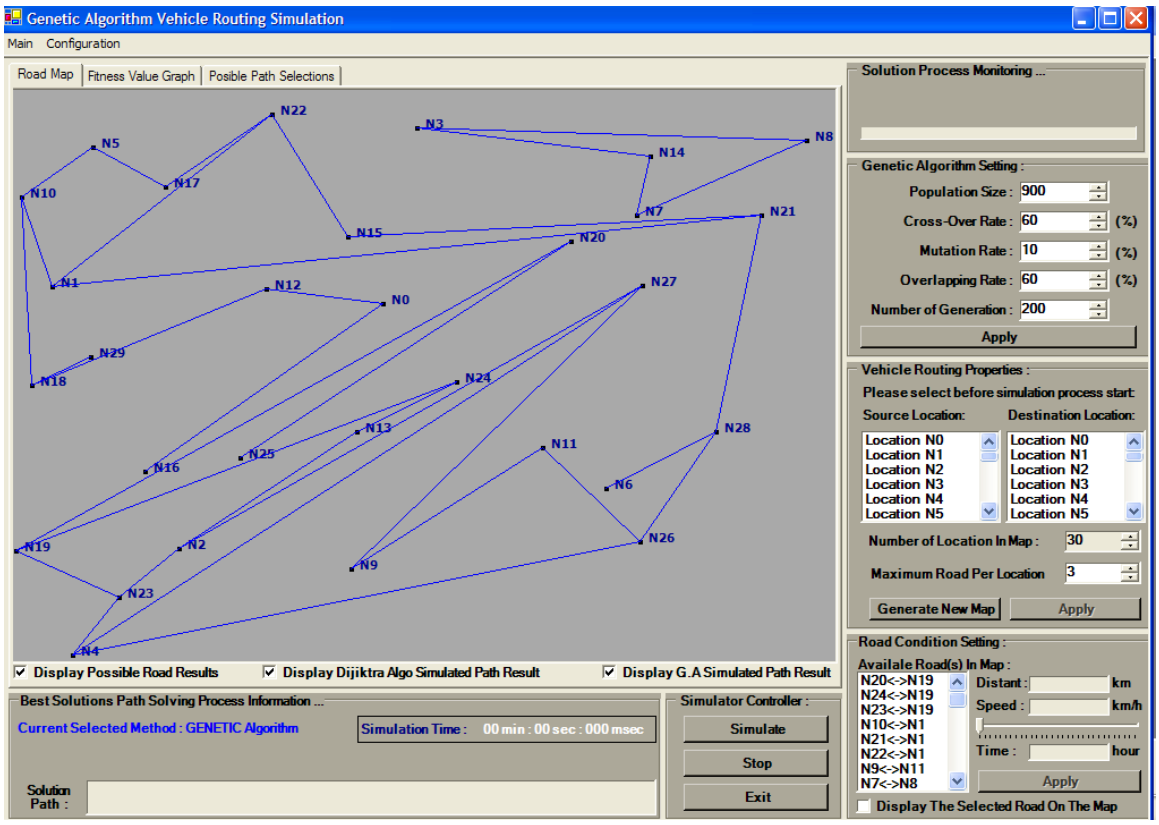

Figure 5.19 : Speed Initial Screen

We shall choose the Source and Destination, Figure 5.20

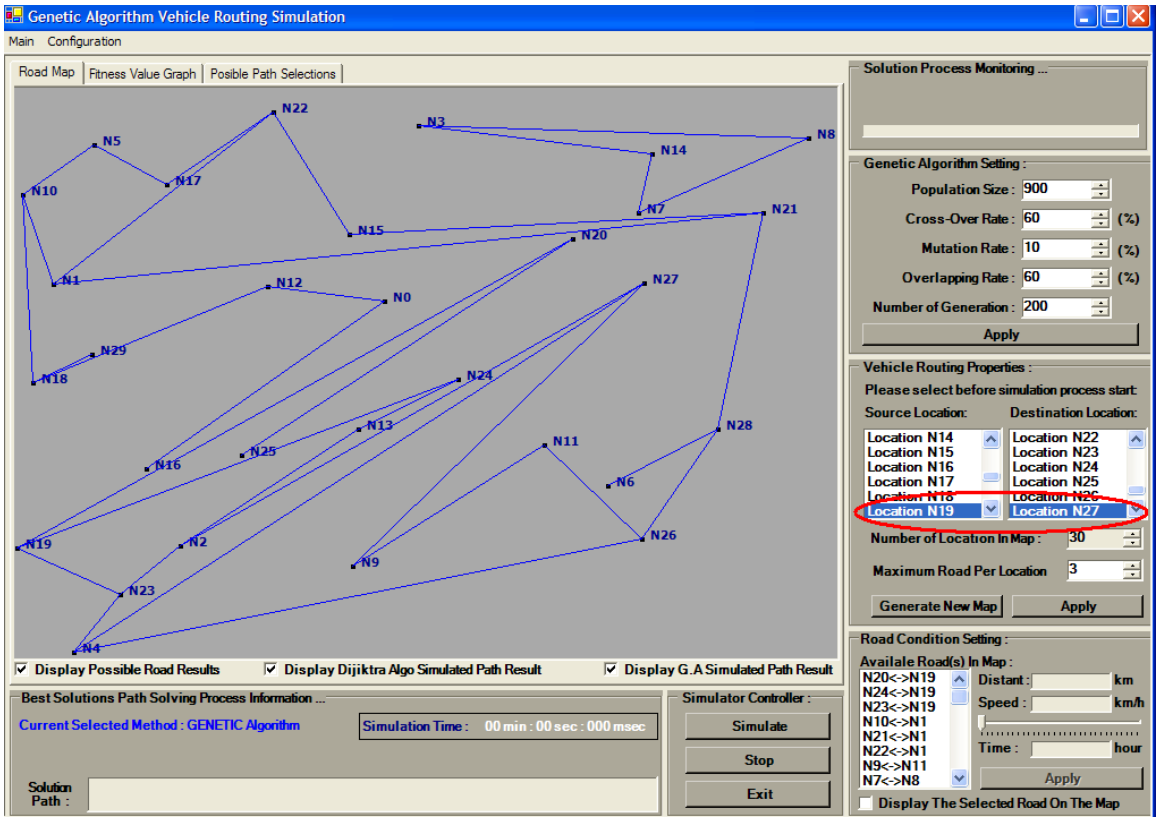

Figure 5.20 : Choose Source & Destination for Speed Setting

Once we click “Apply” the source and destination icons will be displayed on the map as displayed in Figure 5.21

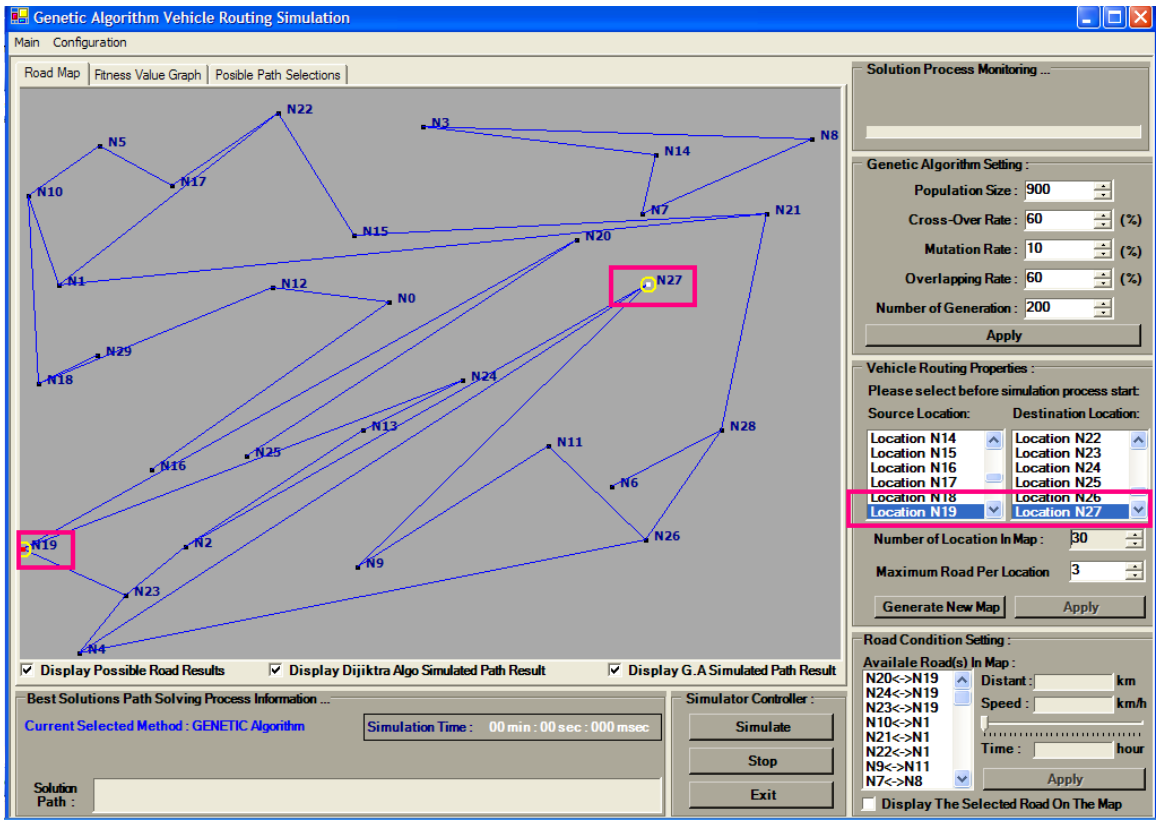

Figure 5.21 : Source & Destination plotted on Map for Speed

When we double click the road on the map, the selected road will appear on the "Available Road(s) In Map" setting. We do not have to scroll down the selection box to locate the desired road. Chosen road will display the current speed that is set on that road. Hence, we can change the road speed as desired.

Cursor is placed and double clicked between Road N19 and N23 and the results in shown in the Road Condition Setting menu, as in Figure 5.22

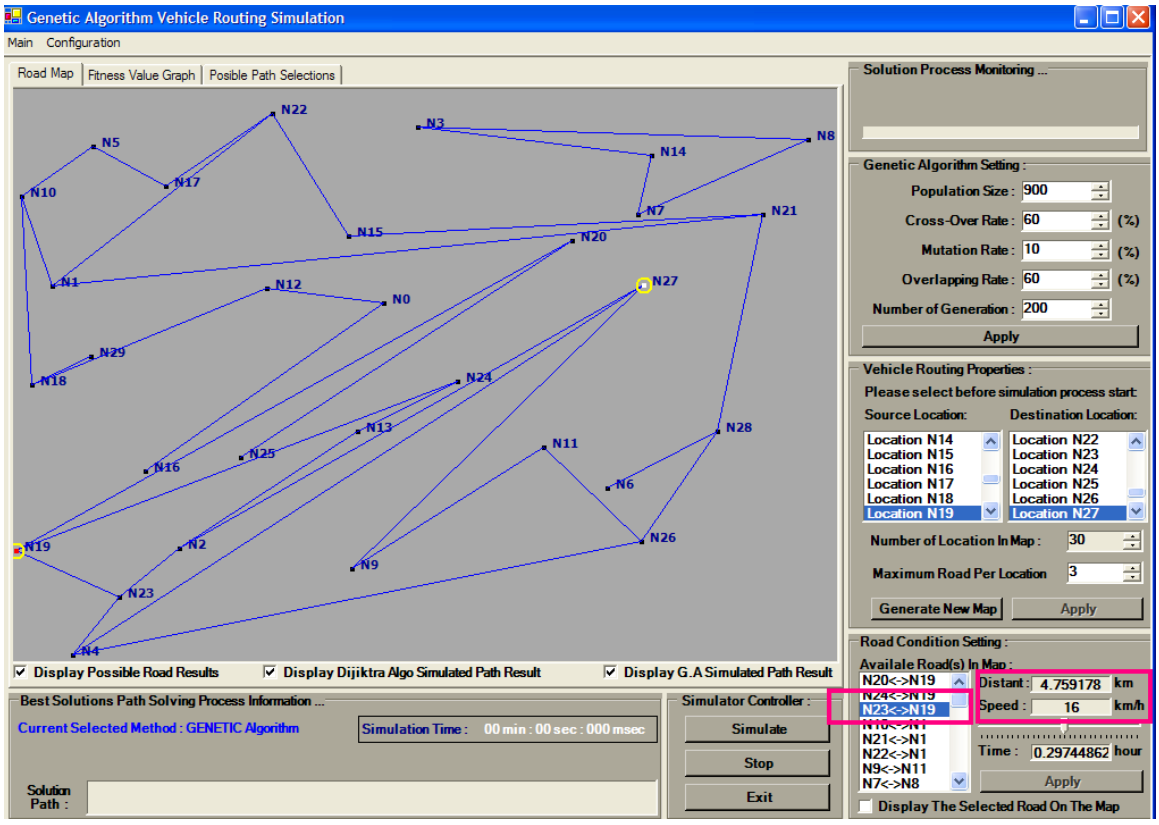

Figure 5.22 : Road Condition Setting for Speed

The default Speed for N19 to N23 is 16 km/h, as seen in Figure 5.22. If we tick on the "Display the selected road on the map" check box, the road chosen will be displayed in Yellow, refer Figure 5.23. Thus, with the same steps as mentioned above, we will get to view the default speed for other roads that are connecting the Source and Destination.

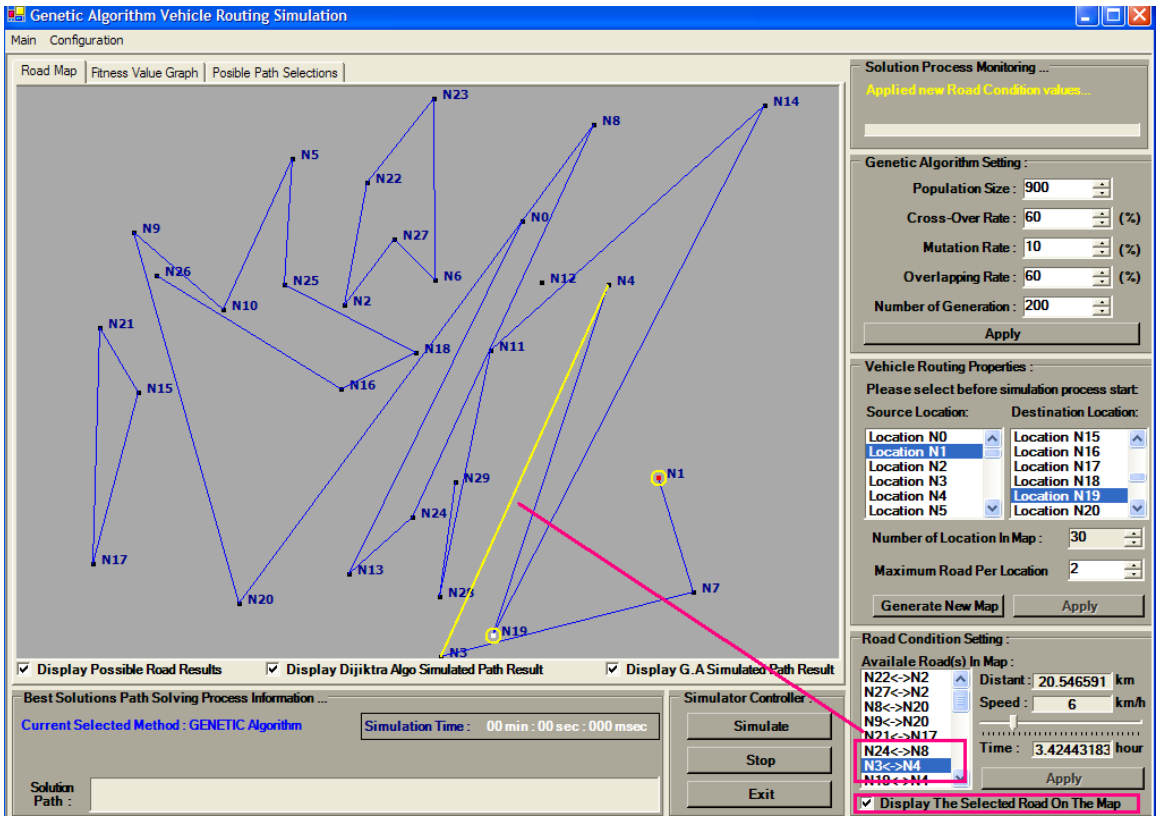

Figure 5.23 : Road for Speed Setting Displayed on Map

5.7.2 Speed Testing

Figure 5.24 shows the current default speed for each road for map in Figure 5.23

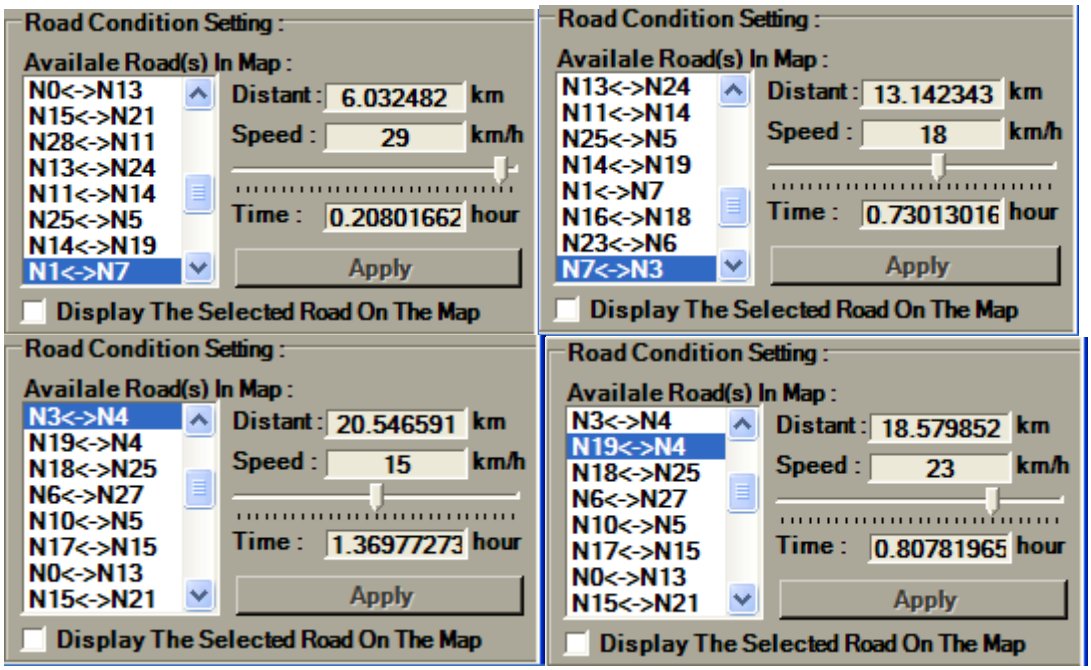

Figure 5.24 : Default Speed Setting for Roads Connection Source & Destination

Now, let's simulate the results.

Simulation Time : 24 sec 125 msec
Distance Travelled : 58.3km
Time Travelled : 3.12h

Then, we change the speed for each road.
Remember to click "Apply" upon changing the road speed. The speed change for each road connecting the Source and Destination is shown in Table 5.8
Results for this changes is shown in table 5.9

Table 5.8 : Speed Change for each road connection Source & Destination

ROAD	N1 – N7	N7 – N3	N3 – N4	N4 – N19
DISTANCE(km)	6.032	13.142	20.547	18.580
SPEED (km/h)	29	18	15	23
SPEED (km/h)	26	15	12	20
SPEED (km/h)	23	12	09	17
SPEED (km/h)	20	09	06	14

5.8 Test Results

5.8.1 Population Size and Number of Generation as Constants

A) Overlapping Rate and Mutation Remains Same

Table 5.9 : Effect of Crossover rate on GA results

Overlapping Rate	50	50	50
Crossover Rate	80	60	40
Mutation Rate	10	10	10
Results	N18-N13-N21-N26-N7-N28-N16 [Dist=64.49km,Time=5.09h]	N18-N13-N21-N16 [Dist=48.74km,Time=3.47h]	N18-N13-N21-N16 [Dist=48.74km,Time=3.47h]

B) Overlapping Rate and Cross over remains same

Table 5.10 : Effect of Mutation on GA results

Overlapping Rate	50	50	50
Crossover Rate	60	60	60
Mutation Rate	10	20	50
Results	N18-N13-N21-N26-N7-N28-N16 [Dist=64.49km,Time=5.09h]	N18-N13-N21-N16 [Dist=48.74km,Time=3.47h]	Can't Get Correct Solution. Perform more generation to obtain solution path!

5.8.2 Number of Generation as Variable

Table 5.11 : Effect on GA results based on No. of Generation

Crossover Rate	60	60	60
Mutation Rate	10	10	10
No Of Generation	200	300	500
Results	N8-N17-N24-N27-N18-N19 [Dist=70.53km, Time=7.87h]	N8-N6-N7-N22-N0-N14-N15-N18-N19 [Dist=101.53km, Time=17.14h]	N8-N6-N7-N22-N19 [Dist=54.01km, Time=6.58h]

5.8.3 GA Computation Time vs Djikstra Computation Time

Table 5.12 : GA Computational Time vs. Djikstra Computational Time for 30 Nodes

Algo Used / Results	**GA 300 Generation**	**GA 500 Generation**	**GA 700 Generation**	**Djikstra**
Computational Time (seconds)	42.546	69.109 (1min 9.109sc)	92.890 (1min 32.890sc)	0.015
Distance (km)	123.61	105.69	76.88	76.88
Travel Time (hours)	14.01	12.88	09.59	09.59

Table 5.13 : Result of Djikstra Computational Time for Increment Nodes

No. Nodes / Results	**3,000**	**5,000**	**8,000**	**10,000**
Computational Time (seconds)	120.478 (D) 90.767 (GA)	240.405 (D) 92.612 (GA)	300.690 (D) 90.800 (GA)	900.998 (D) 91.724 (GA)
Distance (km)	76.88	76.88	76.88	76.88
Travel Time (hours)	09.59	09.59	09.59	09.59

(D) – Djikstra (GA) Genetic Algorithm

5.8.4 Fitness Value vs. Number of Generation

Kindly refer section 5.4 for graphical evidence of the performance. Those graphs are produced by the SHAVRS.

5.8.5 GA Computational Time vs. Full Search

Kindly refer Table 5.6 for tabular illustration of the results.

5.8.6 No. of Successful Search vs. Number of Generation

With reference to section 5.6, table 5.7, here the capability of GA to obtain the results is tested. As mention much earlier in this dissertation, GA can find "good" solution that it approximation of the solution and not necessary the "best" solution.

For each chosen number of generation and population, the simulation is tested for 5 runs, and examines out of the 5 runs, how many runs has produced a successful solution. Table 5.7 would display the results.

5.8.7 Speed Testing

Table 5.14 : Simulation Time & Time Traveled on Speed Change for each Path

ROAD	N1 – N7	N7 – N3	N3 – N4	N19 – N4	Simulation Time	Distance Traveled	Time Traveled
SPEED (km/h)	29	18	15	23	24sec 125msec	58.30km	3.12h
SPEED (km/h)	26	15	12	20	24sec 093msec	58.30km	3.75h
SPEED (km/h)	23	12	09	17	21sec 562msec	58.30km	4.73h
SPEED (km/h)	20	09	06	14	20sec 234msec	58.30km	6.51h

Table 5.15 : Distance Traveled Calculation

Distance Travelled from N1 to N19 for all speed are the same = 58.30km

(n1-n7) + (n7-n3) + (n3-n4) + (n4-n19) = 58.30km
6.032 + 13.142 + 20.547 + 18.580 = 58.30km

Distance between the Source and Destination will not change unless, the simulation has found another new path that connects the Source and Destination. In this case, author has done few simulations and found the shortest path based on genetic algorithm search. Once this shortest path is identified then the speed testing was done.

Table 5.14 shows the Simulation time taken, Time Traveled when road speed is changed. Distance Traveled is a constant, refer table 5.15.

5.9 Test Analysis

5.9.1 Crossover Rate Analysis

Rate that is suitable for crossover is 50 – 60%, above and below this rate would produce results which are not so accurate. It is a bell curve graph.

5.9.2 Mutation Rate Analysis

Mutation cannot go any higher than 10%. The higher the mutation the less likely we are to get a solution. The result might not even connect the source to the destination if mutation is increased to 50% or more.

5.9.3 Number of Generation Analysis

When the generation number is increased, the results appear to be more acute. Result would show a positive linear graph.

5.9.4 Computational Time for Djikstra vs. Genetic Algorithm

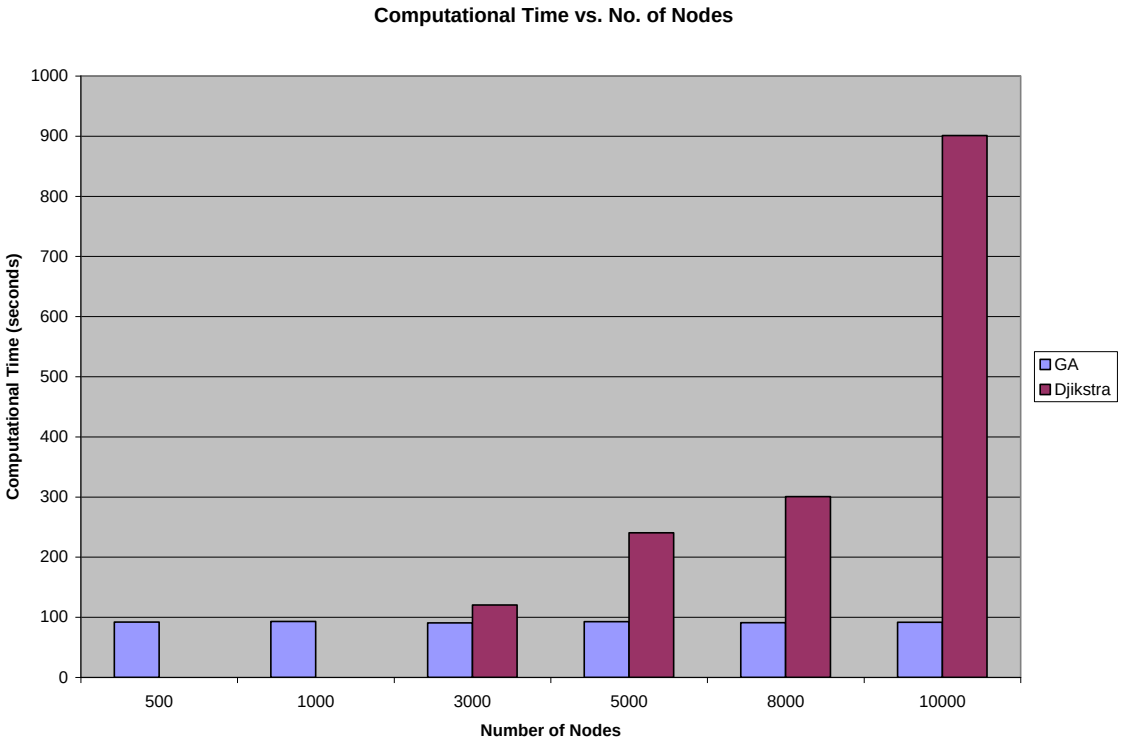

Figure 5.25 : GA Computational Time vs. Djikstra when Nodes Increases

Table 5.16 : GA Computational Time vs. Djikstra when Nodes Increases

No.Of Nodes / Algorithm	500	1,000	3,000	5,000	8,000	10,000
GA	91.985	92.890	90.767	92.612	90.800	91.724
Djikstra	0.015	0.028	120.478	240.405	300.690	900.998

Djikstra will always display the results at a faster computational time compared to Genetic Algorithm when the number of nodes/location is small. The more we increase the number of generations for GA, the results becomes much better in the context of travel time and distance traveled, but it would require longer computational time for processing the number of generation.

When the map becomes more complex size, it is noticed that GA would give a good solution in a minimum time compared to Djikstra. This is seen when the number of nodes on the map increases (when N gets large) to more than 10,000. GA would give a global results where else Djikstra would only get the local minima.

(Chang, 2002 pp.567] has quoted that Djikstra Algorithm is not suitable for large networks or real-time communications, since it has a prohibitive computational cost.

5.9.5 Computational Time for Full Search vs. Genetic Algorithm

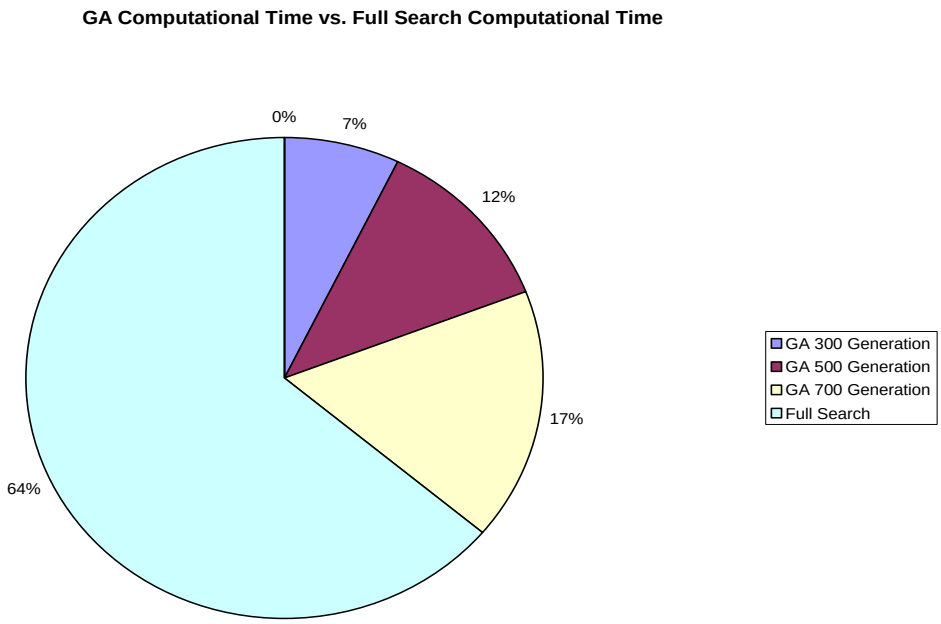

Figure 5.26 : GA Computational Time vs. Full Search Computational Time

Table 5.17: GA Computational Time vs. Full Search Computational Time

Algorithm	GA 300 Generation	GA 500 Generation	GA 700 Generation	Full Search
Time (seconds)	42.531	72.796	101.781	382.265

Full Search searches all possible path for the designated Source and Destination, then provides you the Fittest Value in terms of shortest path, GA algorithm concept works the same but the GA at any point of time would definitely give a better computational time compared to Full Search.

5.9.6 Fitness Value vs. Number of Generation

Fitness value converges to a lower value (stronger solution) when the number of generation increase. Fitness value goes lower (better) when generation increase, therefore Fitness is inversely proportional to Number of Generation.

5.9.7 No. of Successful Search against No. of Generation and Population

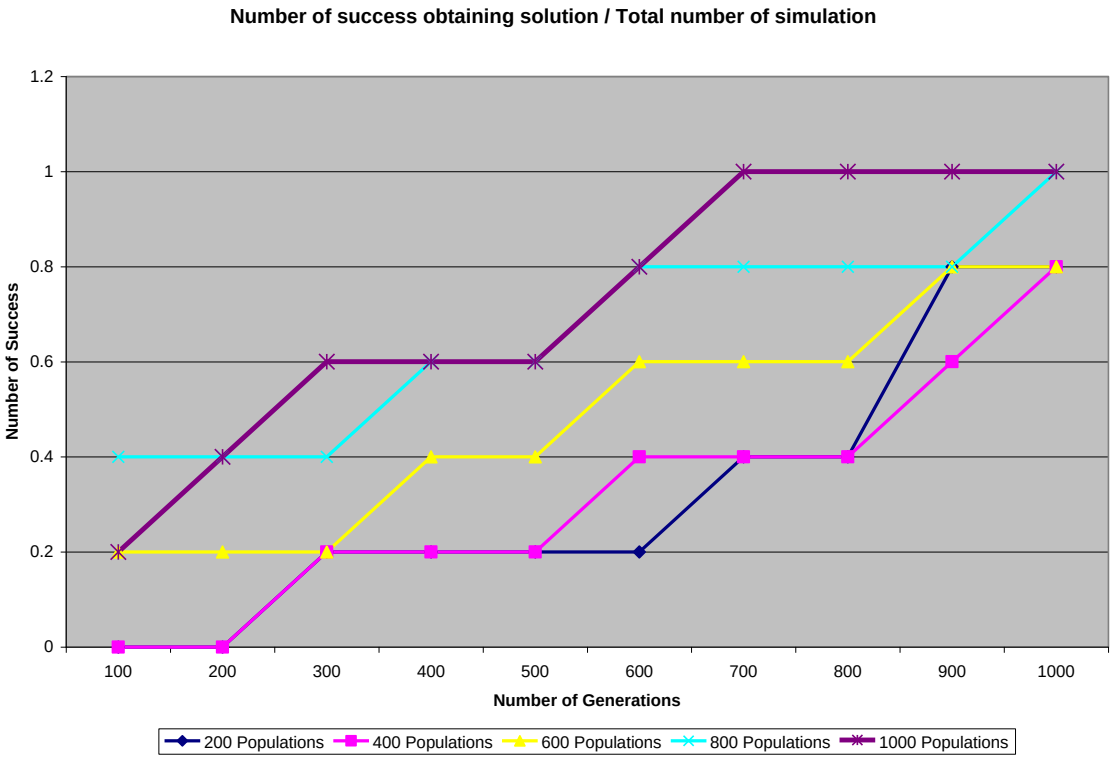

Figure 5.27 : Number of success obtaining solution / Total number of simulation

The bigger the search space and the number of generation would produce chances of a good solution. GA that explicitly evaluates a small number of individuals is implicitly evaluating a much larger group of individuals Due to the parallelism that allows them to implicitly evaluate many schema at once, genetic algorithms are particularly well-suited to solving problems where the space of all potential solutions is truly huge - too vast to search exhaustively in any reasonable amount of time. Most problems that fall into this category are known as "nonlinear". In a linear problem, the fitness of each component is independent, so any improvement to any one part will result in an improvement of the system as a whole. Needless to say, few real-world problems are like this. Nonlinearity is the norm, where changing one component may have ripple effects on the entire system,

and where multiple changes that individually are detrimental may lead to much greater improvements in fitness when combined. Nonlinearity results in a combinatorial explosion: the space of 1,000-digit binary strings can be exhaustively searched by evaluating only 2,000 possibilities if the problem is linear, whereas if it is nonlinear, an exhaustive search requires evaluating 2^{1000} possibilities - a number that would take over 300 digits to write out in full.

5.9.8 Performance & Running Time with and without initialization

The following GA parameters were used :

- Population size : 600
- Probability of mutation : 0.5
- Weights in evaluation function: $W_1 = 5$, $W_2 = 156$, $W_3 = 50$, $W_4 = 25$, $W_5 = 50$
- Number of generations using our clustered initialization method: 40. This was more than sufficient for getting near optimal solutions. We ran for 400generations for the randomly initialized genetic algorithm.

All results are over 10 runs with 10 random seeds:

Performance and running time with and without initialization are compared. Figure 5.18 shows convergence performance on the first benchmark which has fairly uniform time windows. The figure plots the average over ten runs of the best tour length on the y-axis. Both methods converge quickly and result in similar performance with no significant difference. Uniform time windows lead to more feasible solutions and fewer constraints thus negating any initial advantage for our approach.

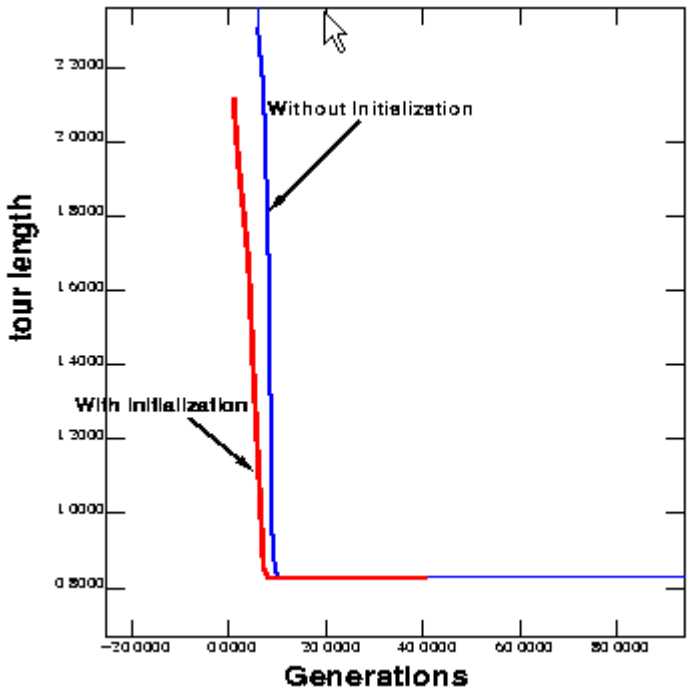

Figure 5.28 : Comparing Convergence with and without Initialization.

Without initialization, convergence takes much longer and even after running for 400 generations, does not produce good a tour as with initialization approach.

In general, non random initialization approach produces better or equivalent results than a randomly initialized genetic algorithm. Furthermore, results are obtained far more quickly. Within 20 generations for our approach versus a few hundred for random initialization on most problem.

Conclusion

This simulation is implemented with a new initialization procedure for genetic algorithms applied to a common version of the vehicle routing problem. This approach is especially effective when customers are located in clusters. In this current implementation,

clustering is done visually. Combined with a novel mutation operator, this simulation approach produces excellent solutions in much less time when compared to randomly initialized genetic algorithms.

5.9.9 Speed evaluation

Speed settings on each path that the vehicle travels from source to reach its destination. Speed on path against the computational time.

Table 5.18 : Speed on each path against Computational Time

Comp.Time / Path	24.13	24.09	21.57	20.23
N1-N7	29	26	23	20
N7-N3	18	15	12	9
N3-N4	15	12	9	6
N4-N19	23	20	17	14

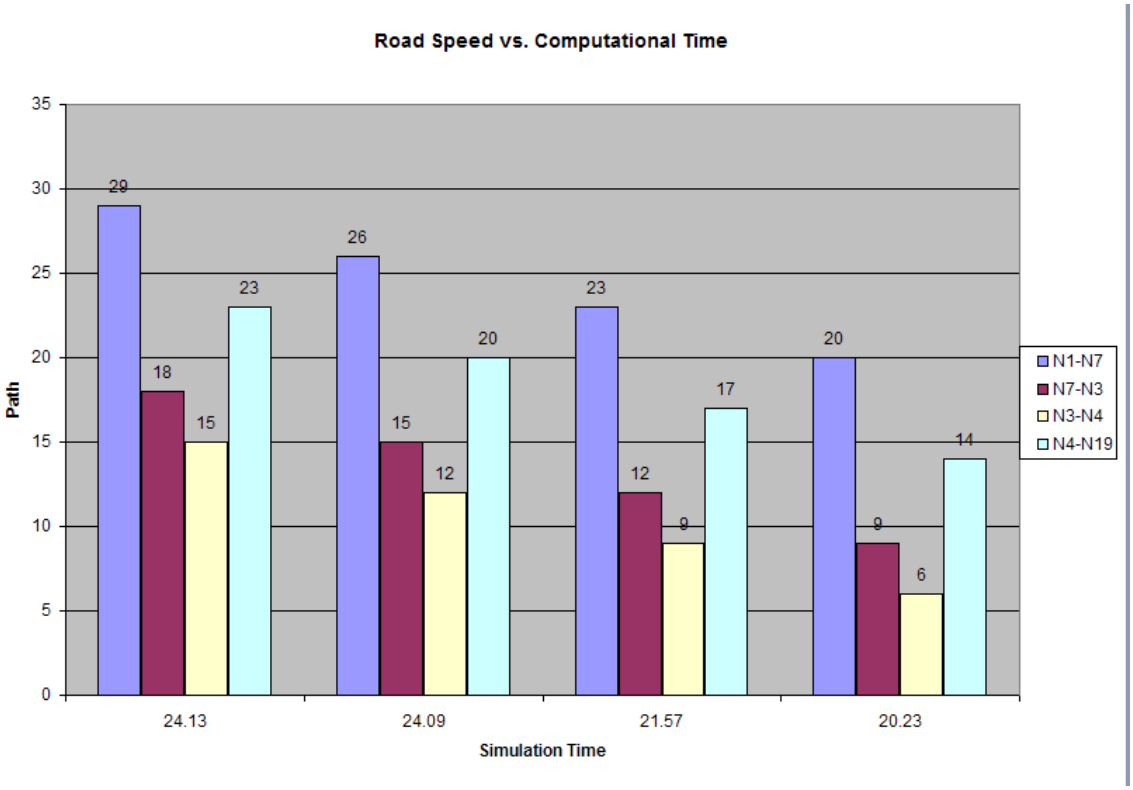

Figure 5.29 : Road Speed vs. Computational Time

Speed on path against the traveling time

Table 5.19 : Speed on each path against Traveling Time

TimeTravel/ Path	3.12	3.75	4.73	6.51
N1-N7	29	26	23	20
N7-N3	18	15	12	9
N3-N4	15	12	9	6
N4-N19	23	20	17	14

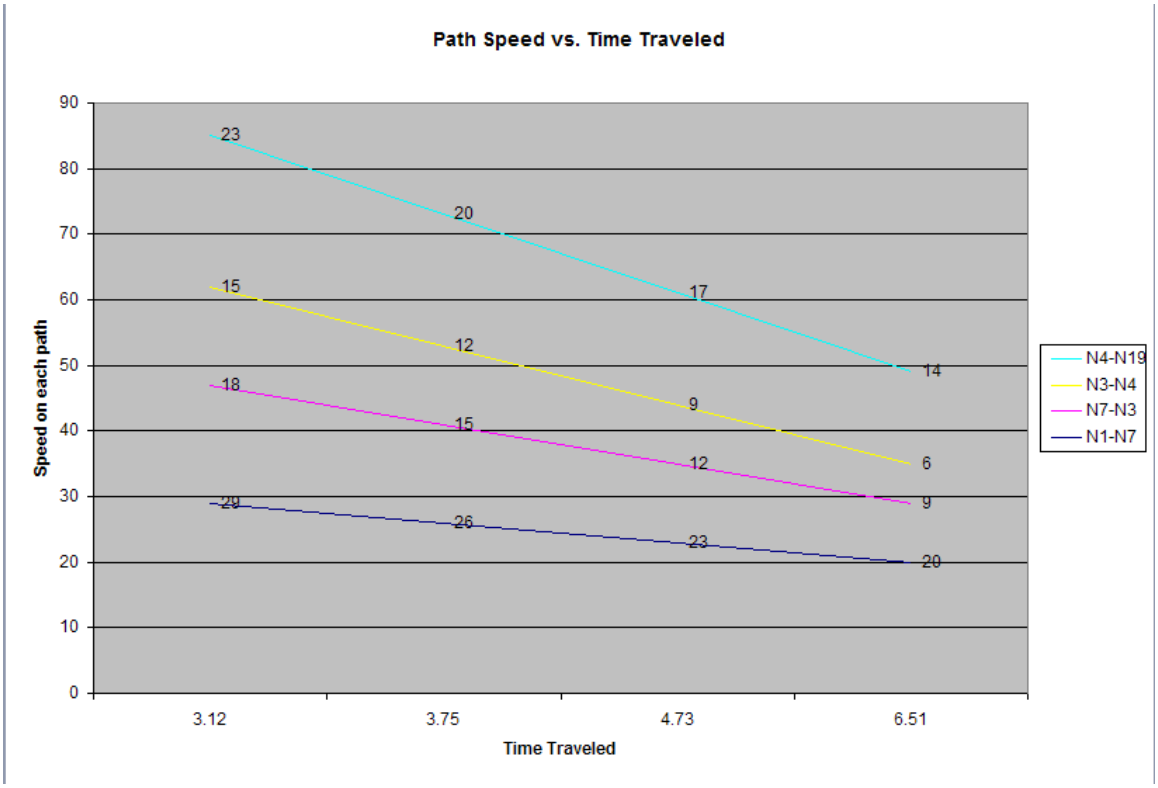

Figure 5.30 : Path Speed vs. Time Traveled

Speed of a vehicle that moves on each path that connects the Source and Destination does make a difference in the computational time and time traveled.
When the speed on each path gets lesser the computational time does go lower but the time travel in increased.

5.10 Discussion

5.10.1 Simulation Constraints

This simulation was developed with some constrains such as :

- The roads in the map does not cross each other
- Number of roads connection to each node in the map is the same.

  Eg. Node A, number of road connected to it, is 3 roads. Therefore, all nodes in the map will be connected to 3 roads.
- The road condition is dynamic.
- Source (first element in the array) and Destination (last element in the array) does not take part in the GA operations. Source and Destination is fixed.

5.10.2 Enhancement on simulation techniques

The simulation can design to be tested with:

- Different GA operators
- Different sets of reproductions operators.
- Different length of array (chromosome)
- Source (first element in the array) and Destination (last element in the array) can take part in the GA operations.
- Null value (-1) in the array can be set not to take part in any GA operations

5.10.3 Observation on Solutions via Genetic Algorithm

There are several general observations about the generation of solutions via a genetic algorithm:

- Unless the fitness function is handled properly, GAs may have a tendency to converge towards local optima rather than the global optimum of the problem.
- Selection is clearly an important genetic operator, but opinion is divided over the importance of crossover versus mutation. Some argue that crossover is the most important, while mutation is only necessary to ensure that potential solutions are not lost. Others argue that crossover in a largely uniform population only serves to propagate innovations originally found by mutation, and in a non-uniform population crossover is nearly always equivalent to a very large mutation (which is likely to be catastrophic).
- GAs can rapidly locate *good* solutions, even for difficult search spaces. For specific optimization problems and problem instantiations, simpler optimization algorithms may find better solutions than genetic algorithms (given the same amount of computation time). GA practitioners may wish to try other algorithms in addition to GAs.
- As with all current machine learning problems it is worth tuning the parameters such as mutation probability, recombination probability and population size to find reasonable settings for the problem class you are working on. A very small mutation rate may lead to genetic drift or premature convergence of the genetic algorithm in a local optimum. A mutation rate that is too high may lead to loss of good solutions.
- How the fitness function is evaluated is an important factor in speed and efficiency of the algorithm.

Typically, GA has been utilized in solving complex optimization problems. GA confer the following advantages:

- They evaluate several solutions simultaneously, covering a large search space.
- They work well in parallel implementation.
- They optimize parameters with very complex cost functions.
- They create a list of optimal solutions, not just a single solution.

In the context of the general optimization problem for routing, a method is needed to compare routes based on their weights to select the shortest path. Genetic methods are well suited to address the route discovery and selection problem.

5.11Chapter Summary

This chapter covers all the testing that had been carried out, results obtained and analysis made based on the results. There are various testing done, such as GA parameters testing to see how population size, number of generation, crossover rate and mutation rate takes influence the GA solution/results, computational time comparison against GA, Full Search and Djikstra, Convergence of fitness value against number of generation and number of successful search per execution. Also, covers alteration to the GA techniques that can be experimented with in the context of chromosome representation, selection mechanism, manner of crossover and mutation, stopping criteria and heuristics which is different from the solution method that was used to develop this SHAVRS system.

Chapter 6

Conclusion

6.1 Conclusion Overview

This research investigates the use of genetic algorithms in vehicle routing. Although genetic algorithms have been previously utilized by other researchers in solving the vehicle routing problem, this research aims to create different a way of chromosome representation and the introduction of overlapping rates to determine the number of chromosomes sent for cross-over and mutation.

6.2 Contribution Highlights

In this study, the potential of using GA to solve the shortest route by using proportional selection Algorithm with two-point crossover and random mutation algorithm is applied. Chromosome representation that was used is fixed length chromosome. In the development, ‘overlapping rate’ was introduced to decide the population split for crossover and mutation respectively. Sorting of the fittest solution is done each time a crossover is completed before moving to mutation, the reasoning for this approach is discussed in detail in chapter 3 (Solution Method). Approach of Pair_Road_Connected_Weight, Road_Not_Connected_Weight and Road Shortening is also introduced refer section 3.3.

There are few primary objectives to this project. First, we have gained understanding of the use of GA in solving the shortest route problem for VRP. Secondly, we have investigated the shortest route problem in three algorithms, Dijkstra, Full Search and GA.

As mentioned such earlier in this report, that Djikstra is merely used to verify the correctness of GA result while Full Search is used to compare the computational time.

For each of the shortest route problem domain, the simulation had been tested on randomly generated road map with different weights settings for each road. The test runs are performed with tuning of the GA parameters as discussed in Chapter 5.

From the testing and results analysis, the GA solution has achieved the results to find the optimum solution which is the shortest route with minimum weight. Weight in this context means travel time and distance traveled.

In conclusion, the goal of this dissertation,, which is to demonstrate the potential of using GA to solve shortest route problem for VRP has been achieved.
New perspective brought to the topic are a new way of chromosome representation, different fitness value formula, allows to determine the speed traversed on each road, road shortening and overlapping rate. calculate

6.3 Future Work

The simulated system developed in this study can be enhanced to produce much better results. For example, future work that could be carried out to further improve the simulation is as listed below.

GA can be implemented with other heuristic properties, selection criteria, a different set of crossover and mutation operators and enhanced assumptions to compare the results of

the route produced in this study. A full VRP problem can be analysed by extending the proposed algorithm for multi destination and routes to end at the depot. This proposed method can also be experimented further with other testing criteria.

For real-world applications it is also necessary that to conduct simulations in which the results of this study are applied to actual maps consequently analyze situations on road congestion (traffic that vary at different time of the day), traffic light (number of traffic lights, its location and timing delay that differs based on traffic time) and road condition (road works which influences traveling time).

References

Al-Tabtabi H. and Alex A. (1998). "An evolutionary approach to the capital budgeting of construction projects." Cost Engineering, AACE, 40(10), 28-34.

Bullnheimer B. , Hartl R. F. , Strauss C. (1997) "Applying the Ant System to the Vehicle Routing Problem" Department of Management Science, University of Vienna.

Chang W. A., *Student Member, IEEE,* and Ramakrishna R. S. *, Senior Member, IEEE (2002) "*A Genetic Algorithm for Shortest Path Routing Problem and the Sizing of Populations", IEEE Transactions On Evolutionary Computation, Vol. 6, No. 6, December, pp. 566-579

Christofides N. , Mingozzi A. , Toth P. (1979) "The Vehicle Routing Problem. In: Combinatorial Optimization. John Wiley", pp. 315–338

Dantzing G. , Ramster R. (1959) "The truck dispatching problem. Management Science 6", pp. 80–91

Dijkstra E. W. (1959) "A Note on two Problems in Connection with Graphs." Numerische Mathematics 1, October, pp. 269-271.

Fisher M. L. , Jornsten K. O. and Mansen O. B. G. (1992) "Vehicle Routing with Time Windows" Working dissertation.

Fogel D. B. (2000) "Evolutionary Computation", IEEE Press, 2nd edition.

Forrest, Stephanie (1993) "Genetic algorithms: principles of natural selection applied to computation.", *Science,* vol.261, pp.872-878.

Garey M. and Johnson D. (1979) "Computers and Intractability: a guide to the theory of NP-Completeness", W.H. Freeman.

Gen M. , Runwei C. , and Dingwei W. (1997) "Genetic Algorithms for solving shortest path problems", IEEE Computer, Vol.30, No.8, pp. 401-406.
Gendreau M. , Potvin J. (1998) "Dynamic Vehicle Routing and Dispatching",
NEC Research Institute.

Glover F. (1986) "Future paths for Integer Programming and Links to Artificial Intelligence. Computers and Operations Research."

Goldberg D. E. (1989) "Genetic Algorithms in Search, Optimization and Machine Learning, Reading", MA, Addison-Wesley.

References

Graham-Rowe, Duncan (2002) "Radio emerges from the electronic soup." *New Scientist*, vol.175, no.2358, p.19 (August 31). Available online at http://www.newscientist.com/news/news.jsp?id=ns99992732.

Haupt R. and Haupt S. E. (1998) "*Practical Genetic Algorithms*", John Wiley & Sons, pp. 17.

Ida K. and Gen M. (1995) "An algorithm for solving bicriteria shortest path problems with fuzzy coefficients", Japanese Journal of Fuzzy Theory and Systems, Vol. 7, No. 1, pp. 142-152.

Mitchell M. (1996) "An Introduction to Genetic Algorithms" MIT Press, Cambridge, MA

Mohamed E. M. , and Saad M. A. E. (2000) "A Genetic Algorithm for Joint Optimization of Capacity and Flow Assignment in Packet Switched Networks", in 17th National Conference, Feb., pp. 1-6, Egypt.

Ravindran A. , Phillips D. T., and Solberg J. J. (1987) "Operations Research: principles and practice", John Wiley & Sons.

Topkis D. M. (1991) "A *k* shortest path algorithm for adaptive routing in communication networks, IEEE Trans. Robot. Automat.", Vol.7, pp. 342-350, June.

Toth P. , Vigo D (2001) "The Vehicle Routing Problem. Monographs on Discrete Mathematics and Applications". SIAM, Philadelphia